Journal of Statistical Software

September 2023, Volume 107, Issue 1. *doi: 10.18637/jss.v107.i01*

Modeling Population Growth in R with the biogrowth Package

Alberto Garre
Wageningen University & Research

Jeroen Koomen
Wageningen University & Research

Heidy M. W. den Besten
Wageningen University & Research

Marcel H. Zwietering
Wageningen University & Research

Abstract

The growth of populations is of interest in a broad variety of fields, such as epidemiology, economics or biology. Although a large variety of growth models are available in the scientific literature, their application usually requires advanced knowledge of mathematical programming and statistical inference, especially when modelling growth under dynamic environmental conditions. This article presents the **biogrowth** package for R, which implements functions for modelling the growth of populations. It can predict growth under static or dynamic environments, considering the effect of an arbitrary number of environmental factors. Moreover, it can be used to fit growth models to data gathered under static or dynamic environmental conditions. The package allows the user to fix any model parameter prior to the fit, an approach that can mitigate identifiability issues associated to growth models. The package includes common S3 methods for visualization and statistical analysis (summary of the fit, predictions, ...), easing result interpretation. It also includes functions for model comparison/selection. We illustrate the functions in **biogrowth** using examples from food science and economy.

Keywords: kinetic modelling, model fitting, predictions, dynamic modeling, R, predictive microbiology, uncertainty.

1. Introduction

The analysis of growth is a research focus of many scientific fields. However, the actual meaning of the term "growth" can vary widely. In biology, growth can refer to an increase in the size of an individual organism or in the number of members in a population due to

replication, whereas in economic studies growth may relate to the change in the monetary value of the goods produced by an economy. Regardless of the field, a usual attribute of growth is that its underlying mechanisms are complex and not fully understood. For instance, the replication of bacterial cells is determined by a large network of biochemical reactions that has not been described completely (Notebaart, Kintses, Feist, and Papp 2018). This complexity also applies to other fields outside of microbiology, such as socio-economic systems. For that reason, growth is usually described using mathematical models at the population level, where the meaning of the term "population" can vary broadly between fields (number of cells, gross domestic product of a country, ...).

Mathematical models describing growth at a population level have many applications. Nonetheless, these can be roughly divided in two categories: prediction and inference. Model prediction refers to the estimation of the future state of a system. For instance, growth models can be used to predict the increase in the population size of a spoilage microorganism, supporting shelf life estimation (García, Vilas, Herrera, Bernárdez, Balsa-Canto, and Alonso 2015). Inference, on the other hand, is related to analyzing the parameters of growth models estimated under different conditions. For instance, changes in the growth rate of a population can be used to assess whether some actions are effective at controlling a disease outbreak.

At the moment of writing this manuscript, several R packages for modeling growth are already available in the Comprehensive R Archive Network (CRAN), although they have some limitations. The packages **grofit** (Kahm, Hasenbrink, Lichtenberg-Fraté, Ludwig, and Kschischo 2010, no longer available in CRAN) and **growthcurver** (Sprouffske and Wagner 2016) include functions for model fitting, but cannot be used for making predictions. On the contrary, **growthmodels** (Perez 2023) can be used for making predictions, but does not include functions for parameter estimation. The package **nlsMicrobio** (Baty, Delignette-Muller, and Siberchicot 2021) is an extension of **nlstools** (Baty, Ritz, Charles, Brutsche, Flandrois, and Delignette-Muller 2015) that defines several growth models from predictive microbiology that can be used for model fitting or making predictions under static conditions. The recent package **growthrates** (Petzoldt 2022) includes functions for both parameter estimation and making predictions, however, this package (as well as the ones mentioned before) cannot include the effect of changes in the environmental conditions on the growth of the population, which can be highly relevant in some case studies. For instance, in the context of food safety, storage temperature is a major limiting factor of microbial growth that varies largely along the food chain, being highly influential on the shelf life of a product (González-Tejedor, Garre, Esnoz, Artés-Hernández, and Fernández 2018). Although it is not specifically designed for modeling microbial growth **drc** includes functions for fitting and making predictions for sigmoid curves that could also be adapted to the description of population growth (Ritz, Baty, Streibig, and Gerhard 2015). On the other hand, this package cannot be used to simulate dynamic environmental conditions.

Outside of the R environment, several software applications have been developed to describe the growth of populations. **BGFit** (Veríssimo, Paixão, Neves, and Vinga 2013) is an online tool that includes several functions for fitting and sharing growth models. **PRECOG** (Fernandez-Ricaud, Kourtchenko, Zackrisson, Warringer, and Blomberg 2016) is also an online application that can estimate various growth parameters of a population. However, it is not based on predictive models, so it cannot be used for making predictions. This is similar to the *pyphe-growthcurves* module of the **Pyphe** Python toolbox (Kamrad, Rodríguez-López, Cotobal, Correia-Melo, Ralser, and Bähler 2020), which can describe growth curves based on

non-parametric methods. **DMFit** is a web tool for model fitting included within **ComBase** (Baranyi and Tamplin 2004) that can be used to fit growth models to data gathered under dynamic environmental conditions.

In this article, we present the **biogrowth** package (version 1.0.3) for R. It includes functions to describe the growth of populations using parametric models at the population level. It implements functions for model fitting and for the calculation of model predictions. One of its main advantages with respect to packages with a similar scope is the inclusion of secondary models to describe the relationship between the growth rate and the environmental conditions. Moreover, this package enables fitting growth models from data gathered under varying environmental conditions conditions, a methodology that is considered by several authors to be superior for parameter estimation (Telen, Logist, Van Derlinden, Tack, and Van Impe 2012; Huang and Li 2020), or the estimation of secondary models from a set of growth rates. Furthermore, the fit can be done using either non-linear regression or an adaptive Monte Carlo algorithm. The growth models (either fitted or user-defined) can be used to make predictions under static or dynamic environmental conditions. A second advantage of **biogrowth** with respect to other approaches is the possibility to include uncertainty in model predictions, a topic whose relevance for microbial risk assessment is nowadays strongly emphasized by regulatory bodies (EFSA Scientific Committee *et al.* 2018; Schendel, Jung, Lindtner, and Greiner 2018). Furthermore, the package allows choosing between different model equations and implements functions for model selection/comparison (visualization and statistical indexes). The package defines S3 classes as the output of the main functions. This type of R classes is often included in this type of package (many of the packages listed above include them) to facilitate the inspection of the model fits and predictions using S3 methods based on generic functions (such as `plot()` or `summary()`). Considering these features, we believe that **biogrowth** is a step forward with respect to the other R packages and other applications already available for modeling population growth.

The **biogrowth** package follows the modeling approach of predictive microbiology, where models are defined in two steps: primary models and secondary models (Buchanan and Whiting 1996). In this approach, primary models describe the relationship between the population size and the elapsed time. Therefore, they have a single explanatory variable (time). In most cases, the maximum specific growth rate of the primary model depends on the environmental conditions. A typical example is the relationship between the maximum specific growth rate of microbial populations and temperature. In predictive microbiology, this type of relationship is described using secondary models. Although this approach was initially intended for describing microbial populations, its principles are applicable to a large number of scientific fields. For instance, some of the models included in **biogrowth** have been used in biotechnology (Gonzales, Kim, and Kim 2019; Martínez-Hernández, Taboada-Rodríguez, Garre, Marín-Iniesta, and López-Gómez 2021), ecology (Jiang *et al.* 2018), medicine (Adam and Bellomo 2012), astrobiology (Spada and Melini 2020), economics (Tsai 2013), operations research (Cabecinhas *et al.* 2018), or social sciences (Gupta, Kumar, Sangam, and Karisiddappa 2002). Therefore, we consider that the functions included in **biogrowth** can be applied to a large variety of fields with (in some cases) minor modifications, as illustrated in the case studies below.

The following section of this paper describes the modeling approach used in **biogrowth** to describe the growth of populations. Next, Section 3 describes the software architecture, presenting the main functions and classes of the package, as well as the mathematical methods

used for model fitting and prediction. This is followed by Section 4 illustrating four common ways to build and apply growth models: based on parameters from the literature, by fitting primary growth models, by fitting secondary growth models, or by fitting both primary and secondary growth models to data gathered under dynamic conditions. The article finalizes with a section summarizing the functions included in **biogrowth** and contextualizing its use in a variety of fields. For reasons of space, not every aspect of the package is described in full detail. The interested reader is referred to the package manual and vignettes for an in-depth description.

2. Models of population growth

2.1. Growth under constant environmental conditions: Primary models

Version 1.0.3 of **biogrowth** implements five primary growth models in algebraic form: the logistic growth model, the Richards model, the modified Gompertz model (Zwietering, Jongenburger, Rombouts, and Van 't Riet 1990), the Baranyi model (Baranyi and Roberts 1994), and the triphasic linear model (Buchanan, Whiting, and Damert 1997). In principle, these model equations are intended to describe population growth under constant environmental conditions. These models consider a sigmoidal relationship between the logarithm of the population size and the elapsed time, a shape commonly observed in a variety of fields. To ease result interpretation, the growth curve is divided in three phases, as illustrated in Figure 1. Even in conditions suitable for growth, it is common for populations to need an adaptation time before growth onset. The duration of this time defines the *lag phase*. Once the lag phase has finished, the population size grows exponentially during the *exponential phase*. Nevertheless, in most situations the population cannot grow indefinitely (e.g. due to space or nutrient constraints). Hence, the moment when the population size becomes stable marks the beginning of the *stationary phase*.

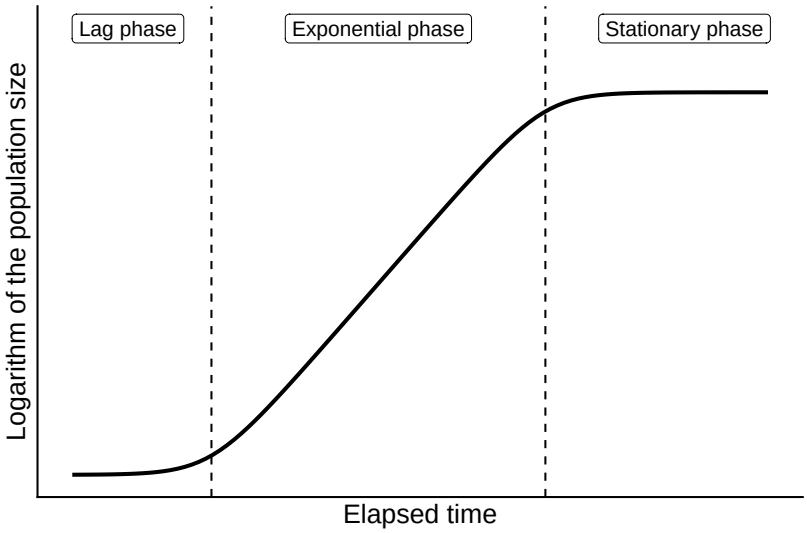

Figure 1: Illustration of a typical growth curve.

The modified Gompertz model (Zwietering *et al.* 1990) is defined in Equation 1, where t is the elapsed time, and $N(t)$ is the size of the population at time t. This model is defined by four model parameters: the logarithm of the population size at $t = 0$ ($\log_{10} N_0$), the duration of the lag phase (λ), the slope of the growth curve during the exponential phase (μ) and the difference between $\log_{10} N_0$ and the logarithm of the maximum population size in the stationary phase (C).

$$\log_{10} N(t) = \log_{10} N_0 + C \left(\exp \left(- \exp \left(\frac{e \cdot \mu}{C} (\lambda - t) + 1 \right) \right) \right) \tag{1}$$

Note that through this article we follow a common convention in predictive microbiology where μ is defined as the slope of the growth curve during the exponential phase in the $\log_{10} N$ versus t plot. This value is different from the specific growth rate (the slope in the $\ln N$ vs t plot), which can be calculated by multiplying μ by $\ln(10)$. By default, the functions in **biogrowth** also follow this convention. Nonetheless, the package includes arguments to make the calculations using either scale (described in Section 3 and in the relevant vignette).

The logistic model is defined in Equation 2, where the variables and parameters have the same interpretation as in the modified Gompertz model.

$$\log_{10} N(t) = \log_{10} N_0 + \frac{C}{1 + \exp \left(\frac{4\mu}{C} (\lambda - t) + 2 \right)} \tag{2}$$

The Richards model is defined in Equation 3. This model has an additional parameter with respect to the modified Gompertz and logistic models (ν), which defines the sharpness of the transition between the different growth phases.

$$\log_{10} N(t) = \log_{10} N_0 + C \left[1 + \nu \cdot \exp \left(1 + \nu + \frac{\mu}{C} (1 + \nu)^{1+1/\nu} (\lambda - t) \right) \right]^{-1/\nu} \tag{3}$$

Regarding the Baranyi model, **biogrowth** uses the algebraic solution under static conditions by (Öksüz and Buzrul 2020). This model uses $\log_{10} N_{\max}$ (the maximum population size) instead of C. The remaining model parameters are the same as the ones in the modified Gompertz model.

$$\log_{10} N(t) = \log_{10} N_{\max} + \log_{10} \frac{1 + 10^{\mu(t-\lambda)} - 10^{-\mu\lambda}}{10^{\mu(t-\lambda)} - 10^{-\mu\lambda} + 10^{(\log_{10} N_{\max} - \log_{10} N_0)}}$$

The triphasic linear model was suggested as a simplified version of other growth models (Buchanan *et al.* 1997; Zwietering, De Wit, Cuppers, and Van 't Riet 1994), where the sigmoidal curve is described using three straight lines. It has the same parameters as the Baranyi growth model.

$$\log_{10} N(t) = \begin{cases} \log_{10} N_0 & \text{if } t < \lambda \\ \log_{10} N_{\max} & \text{if } t > t_{\text{stat}} \\ \log_{10} N_0 + \mu(t - \lambda) & \text{otherwise} \end{cases}$$

with $t_{\text{stat}} = (\log_{10} N_{\max} - \log_{10} N_0)/\mu + \lambda$ standing for the time required to reach the stationary phase.

2.2. Growth under dynamc environmental conditions: Secondary model

The description of population growth accounting for the effect of changes in the environmental conditions requires the definition of the primary growth model as a differential equation. For this, **biogrowth** uses the Baranyi growth model (Baranyi and Roberts 1994). This model is an extension of the exponential growth model that accounts for the delay in the onset of population growth in the lag phase and the saturation observed in the stationary phase. The model initially proposed by Baranyi and Roberts has been simplified during the last decades, resulting in the equation shown below (Perez-Rodriguez and Valero 2012).

$$\begin{cases} \frac{dN}{dt} = \ln(10) \cdot \mu \cdot N(t) \\ \mu = \alpha \cdot \mu_{\text{opt}} \cdot \beta \\ \alpha = \frac{Q(t)}{1+Q(t)} \\ \beta = 1 - \frac{N(t)}{N_{\text{max}}} \\ \frac{dQ}{dt} = \mu_{\text{opt}} \cdot Q(t) \end{cases} \tag{4}$$

In the Baranyi model, the lag phase is modeled based on the hypothesis of an ideal substance ($Q(t)$), which must be produced before exponential growth begins. This is implemented by coefficient $\alpha(t) = Q(t)/(1 + Q(t))$, which acts as a scaling factor of the specific growth rate. The duration of the lag phase is determined by the initial value of $Q(t)$ (Q_0) and the growth rate according to the identity

$$\lambda = \frac{\ln(1 + 1/Q_0)}{\mu_{\text{opt}}}$$

In this model, the stationary phase is introduced with the coefficient $\beta(t)$, which takes the form proposed by (Verhulst 1838). Note that when $\alpha = 1$ and $\beta = 1$, the Baranyi model is reduced to the exponential growth model.

The **biogrowth** package models the impact of the environmental conditions on the growth rate using the so-called *gamma concept* (Zwietering, Wijtzes, De Wit, and Van 't Riet 1992), also called functional response or dimensionless moderators elsewhere. This modeling approach considers that each environmental factor acts as an independent scaling factor for μ with respect to the value of μ observed under optimal growth conditions (μ_{opt}). This can be described for k environmental conditions using the following equation

$$\mu(t) = \mu_{\text{opt}} \cdot \gamma_1(X_1(t)) \cdot \cdots \cdot \gamma_n(X_n(t)); i \in 1, \ldots, k \tag{5}$$

where $\gamma_i(X_i)$ is the "gamma factor" corresponding to environmental condition X_i. In this approach, $\gamma_i(X_i) \in [0, 1] \forall X_i$, so each environmental factor scales down the growth rate with respect to the maximum one (μ_{opt}).

The **biogrowth** package implements three different gamma models: the cardinal parameter model (CPM), a generalization of those proposed by Zwietering and the (adapted) full Ratkowsky model. The CPM (Rosso, Lobry, Bajard, and Flandrois 1995) is described by the following equation

$$\gamma(X) = \frac{(X - X_{\text{max}})(X - X_{\text{min}})^n}{(X_{\text{opt}} - X_{\text{min}})^{n-1}((X_{\text{opt}} - X_{\text{min}})(X - X_{\text{opt}}) - (X_{\text{opt}} - X_{\text{max}})(n-1)(X_{\text{opt}} + X_{\text{min}} - nX))}$$

where X_{min}, X_{opt} and X_{max} are the minimum, optimum and maximum values of X for the population growth. Note that X_{min} and X_{max} usually do not exactly match the values

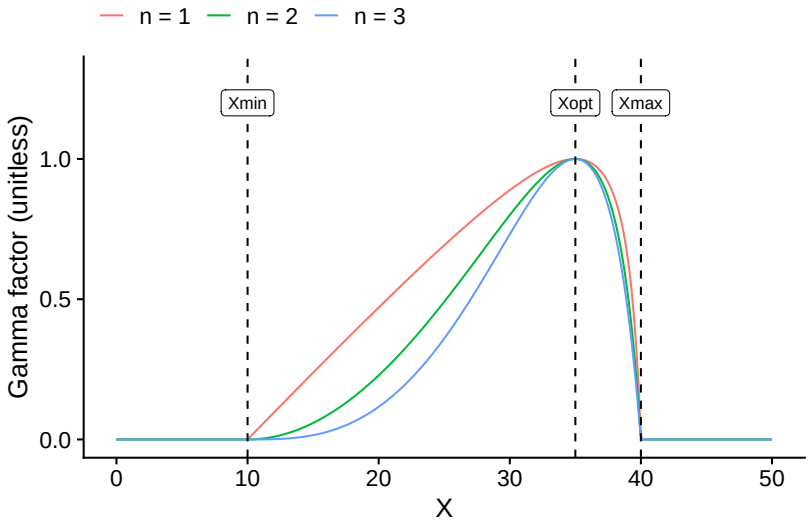

Figure 2: Gamma factor according to the cardinal parameter model (CPM) for an arbitrary environmental factor, X. The lines represent the effect of changing the shape factor (n), while the remaining parameters are fixed: $X_{\min} = 10$ (minimum value for growth), $X_{\text{opt}} = 35$ (optimum value for growth), $X_{\max} = 40$ (maximum value for growth).

observed in growth/no-growth experiments of microbial populations. For that reason, in predictive microbiology, they are often called "theoretical" minimum and maximum values for growth.

The CPM model has a concave shape, whose curvature is defined by parameter n (called "the order" of the cardinal model). Figure 2 illustrates the shape of this model for different values of n.

The secondary models proposed by Zwietering (Zwietering *et al.* 1992) can be expressed as

$$\gamma(X) = \left(\frac{X - X_{\min}}{X_{\text{opt}} - X_{\min}} \right)^n$$

where $X_{\min}$, X_{opt} and n have the same interpretation as in the CPM model.

The Ratkowsky model (Ratkowsky, Lowry, McMeekin, Stokes, and Chandler 1983) was proposed as an extension of an earlier model by the same laboratory (Ratkowsky, Olley, McMeekin, and Ball 1982) with broader validity range. It is defined by the following equation

$$\sqrt{\mu(X)} = b(X - X_{\min}) \left(1 - e^{c(X - X_{\max})}\right)$$

where $X_{\min}$ and $X_{\max}$ have the same interpretation as in the CPM model. Parameters b and c describe the slope of the model for values of X below and above X_{opt}, respectively. However, the output of this model, as formulated by the original authors, can take values outside of $[0, 1]$. Hence, it must be modified before it can be used as a gamma factor. For that, we can calculate the value of X_i for which $\sqrt{\mu(X)}$ is maximum (X_{opt}). By setting $\frac{\partial \mu}{\partial X} = 0$, and considering $c > 0$, we get

$$X_{\text{opt}} = \frac{W_n \left(e^{-cX_{\min} + cX_{\max} + 1}\right) + cX_{\min} - 1}{c}$$

where W_n is the Lambert-W function, defined as the inverse of $f(w) = w \cdot e^w$.
Then, the gamma function corresponding to the Ratkowsky model (γ_{Rtk}) can be calculated as

$$\gamma_{\text{Rtk}} = \left(\frac{\sqrt{\mu}(X)}{\sqrt{\mu}(X_{\text{opt}})} \right)^2 = \left(\frac{(X - X_{\text{min}}) \left(1 - e^{c(X - X_{\text{max}})}\right)}{(X_{\text{opt}} - X_{\text{min}}) \left(1 - e^{c(X_{\text{opt}} - X_{\text{max}})}\right)} \right)^2$$

This function is defined within $[0, 1]$, so it can be used as a gamma function. Note that the scale parameter b fades from the equation, so the model has three parameters: X_{min}, X_{max} and c.

3. Description of the biogrowth package

3.1. Installation

The **biogrowth** package (Garre, Koomen, Den Besten, and Zwietering 2023) is developed in R and available from CRAN at https://CRAN.R-project.org/package=biogrowth. Hence, it can be installed in any computer with R (> 2.1.0) with the command:

```
R> install.packages("biogrowth")
```

The development version of the package can be installed from GitHub using **devtools** (Wickham, Hester, Chang, and Bryan 2022b) with the following command:

```
R> devtools::install_github("albgarre/biogrowth")
```

These commands also install the dependencies of **biogrowth**. The code is written according to the *tidy* philosophy, so it imports most packages from **tidyverse** (Wickham *et al.* 2019). Visualizations are made in **ggplot2**, using also **cowplot** (Wilke 2020) for some visualizations that require subplots. Differential equations are solved numerically using **deSolve** (Soetaert, Petzoldt, and Setzer 2010), and model fitting is done using **FME** (Soetaert and Petzoldt 2010). Finally, **MASS** (Venables and Ripley 2002) and **lamW** (Adler 2023) are used for some statistical calculations. The functions are thoroughly documented using **roxygen2** (Wickham, Danenberg, Csárdi, and Eugster 2022a), and several vignettes have been prepared using **knitr** (Xie 2015). The guidelines of **lifecycle** have been followed to describe the maturity of the different functions (Henry and Wickham 2022).

3.2. Software architecture and main functions

The **biogrowth** package follows a software architecture based on functional programming. Each main function returns an instance of an S3 class, which implements relevant S3 methods to facilitate posterior analyses (`print()`, `summary()`, `plot()`, `predict()`, ...). Table 1 illustrates the main functions implemented in **biogrowth**, as well as their associated S3 classes. They can be divided in three groups: prediction, fitting and analysis. The following sections describe the numerical methods used in the package for each type of calculation, pointing out the main functions and arguments. However, for reasons of space, not every function argument is described in detail. The reader is referred to the function documentation and the package vignette for a thorough description.

Function name	S3 class	Description
`predict_growth()`	'`GrowthPrediction`'	Growth predictions
`fit_growth(approach = "single")`	'`GrowthFit`'	Model fitting to individual experiments
`fit_growth(approach = "global")`	'`GlobalGrowthFit`'	Model fitting to several experiments
`fit_secondary_growth()`	'`FitSecondaryGrowth`'	Fitting secondary growth models
`predict_growth_uncertainty()`	'`GrowthUncertainty`'	Predictions with parameter uncertainty
`predictMCMC()`	'`MCMCgrowth`'	Predictions with parameter uncertainty from fitted models
`compare_growth_fits()`	'`GrowthComparison`'	Model comparison
`compare_secondary_fits()`	'`SecondaryComparison`'	Model comparison for secondary models
`time_to_size(type = "discrete")`	'`double`'	Time to reach a target size
`time_to_size(type = "distribution")`	'`TimeDistribution`'	Distribution of times to reach a target size

Table 1: Summary of main functions and associated S3 classes in **biogrowth**.

3.3. Numerical methods for growth models based on algebraic equations

Model predictions

Calculations under constant environmental conditions are calculated using `predict_growth()` setting `environment = "constant"`. The growth model is passed as a list to `primary_model`, defining the primary model equation and the values of the model parameters. The primary models in **biogrowth** have an algebraic solution, so the prediction is calculated by substituting the values of the model parameter into the equation. Nonetheless, it is calculated only at discrete time points (passed as a numeric vector to `times`). Hence, the precision of posterior analysis (e.g. visualizations or when calculating the time to reach a given size) can be low if `times` is not dense enough.

The function returns an instance of '`GrowthPrediction`' with the results of the calculation. It is a subclass of '`list`' that provides direct access the results of the calculation. It also implements S3 methods for `print()`, `plot()`, `summary()` and `coef()` to facilitate analysis of the prediction.

Predictions with parameter uncertainty

The function `predict_growth_uncertainty()` can include parameter uncertainty in the model predictions. Calculations are done by forwards uncertainty propagation based on Monte Carlo simulation and parametric bootstrap, an approach commonly used in predictive microbiology (Vásquez, Busschaert, Haberbeck, Uyttendaele, and Geeraerd 2014; Garre, Fernández, Lindqvist, and Egea 2017).

The distribution of the model parameters is defined using the `pars` argument. It is a `tibble` (or `data.frame`) with four columns (`par`, `mean`, `sd` and `scale`), where each row describes the marginal distribution of a single parameter. The function uses a normal distribution, although it can also be defined in log- or square-root-scales for each parameter using the `scale` column. Note that the `predictMCMC()` function allows nonparametric calculations based on the Markov Chain of a model fitted using a Monte Carlo algorithm. Moreover, one of the package vignettes describe how to make this calculations for custom parameter distributions.

The calculations are done in two steps. First, the function generates a parameter sample of the size defined by the argument `n_sims`, in the selected scale using `MASS:mvrnorm()`. By default, the function does not consider any parameter correlation, although a correlation matrix can be passed to the `cor` argument. Once the parameter sample has been generated, it is converted to the original parameter scale before calling `predict_growth()` for each sampled parameter vector. This generates a family of curves that estimate the distribution of the population size for the time points defined in the `times` argument.

The results are returned as an instance of '`GrowthUncertainty`'. Although this subclass of '`list`' provides direct access to the results of the calculation, it also implements S3 methods for `print()` and `plot()` to ease the visualization and analysis of the results.

Fitting primary growth models

Primary growth models are fitted to experimental data with `fit_growth()`. The data must be defined as `tibble` (or `data.frame`) with two columns: one defining the elapsed time and a second one with the logarithm of the population size. By default, they must be named "time" and "logN", although this can be changed using the `formula` argument.

The fitting procedure is based on `FME::modFit()`. For each candidate parameter vector, residuals of the log-population size are calculated using `FME::modCost()`, based on the prediction of `predict_growth()`. By default, `fit_growth` uses the default settings of `modFit()`, although they can be modified by passing additional arguments through the `...` argument.

The `modFit()` function uses regression, an algorithm that requires initial guesses for every model parameter. These are passed as a numeric vector to the `start` argument. The **biogrowth** package implements two functions to facilitate their definition: `check_growth_guess()` and `make_guess_primary()`. The former generates a plot comparing the initial guess against the experimental data. The latter applies some heuristic rules to obtain an initial guess of the model parameters. Namely, $\log N_0$ is taken as the minimum population in the data. The maximum population value in the data is used as a guess for $\log N_{\max}$, whereas C is the difference between both values. The value of λ is estimated as the first time point where the population is larger than $\log N_0$ by more than 0.5 log units. The time to reach the stationary phase ($t_{\max}$) is calculated in a similar way, so an initial guess for μ is calculated by $\mu = (\log N_{\max} - \log N_0)/(t_{\max} - \lambda)$. For ν, the function always returns an initial guess of 1 (equivalent to a logistic growth model).

One common issue when fitting sigmoidal growth models to data is poor identifiability. For instance, experimental growth curves usually lack a stationary phase because the experiment ended before the population reached that phase. It has been suggested that fixing some of the model parameters can resolve many identifiability issues in this type of model (Vilas, Arias-Mendez, Garcia, Alonso, and Balsa-Canto 2018). For this reason, `fit_growth()` enables fixing any model parameter prior to model fitting using the `known` argument. Nonetheless, users are advised to use these models with care, as fixing models to arbitrary values may bias parameter inference and result in bias predictions (Schmidt, Emelko, and Thompson 2019).

The model fitted is returned as an instance of '`GrowthFit`', a subclass of '`list`'. It implements a number of S3 methods to analyze the model fitted and extract the statistical properties of the fit. Namely, it implements methods for `print()`, `coef()`, `summary()`, `predict()`, `residuals()`, `vcov()`, `deviance()`, `fitted()`, `logLik()`, `AIC()` and `plot()`.

Fitting secondary growth models

The **biogrowth** package allows fitting secondary growth models to a dataset of maximum growth rates observed under different (constant) environmental conditions with the function `fit_secondary_growth()`. The data must be a `tibble` (or `data.frame`) where each row reports the value of μ observed for one experiment under constant environmental conditions. One column of the data must contain the value of μ observed in the experiment. By default, this column must be named `mu`, although this can be changed using the `formula` argument. Then, additional columns of the data define the value of the environmental conditions of each experiment. The `fit_secondary_growth()` function applies the gamma approach, so it does not impose any limit to the number of environmental factors to include in the model. The model equations assigned to each environmental factor are defined as a named character vector using `sec_model_names`.

Model fitting is based on the `modFit()` function from **FME**. Residuals are calculated using `FME:modCost()` by solving the algebraic solution of the selected secondary model. By default, the default settings of `modFit()` are used. This behavior can be changed using the `...` argument. It is common in predictive microbiology to calculate the residuals of the square root-transformed growth rates to stabilize the variance. However, because the package is intended for a broader audience, `fit_secondary_growth` gives the opportunity to calculate the residuals on three different scales (square root, log-transformed, or no-transformation) using the `transformation` argument.

Fitting secondary models also suffers often from identifiability issues (especially when accounting for numerous environmental factors). This problem can be partly mitigated using the `known_pars` argument to fix any number of model parameters to known values before model fitting. The remaining model parameters require the definition of initial values, passed as a numeric vector to `starting_point`. In order to facilitate their definition, **biogrowth** includes `make_guess_secondary()`, which follows some heuristic rules to generate initial guesses for every model parameter. Namely, for each environmental factor, $X_{\min}$ and $X_{\max}$ are assigned to the minimum and maximum values in the data. The maximum value of μ is taken as the initial guess of μ_{opt} and the value of X where μ is maximum is taken as initial guess for X_{opt}. Finally, an initial guess of 2 is always assigned to n.

The function returns an instance of '`FitSecondaryGrowth`'. This subclass of '`list`' implements several S3 methods to ease the analysis of the model fitted: `print()`, `coef()`, `summary()`, `predict()`, `residuals()`, `vcov()`, `deviance()`, `fitted()`, `logLik()`, `AIC()` and `plot()`.

3.4. Numerical methods for growth models based on differential equations

Model predictions

Passing `environment = "dynamic"` to `predict_growth()` accounts for the effect on the growth rate of changes in the environmental conditions during the experiment. In this case, the growth curve is calculated by solving the differential equation of the Baranyi model (Equation 4) substituting $\mu(t)$ by the selected secondary model according to the gamma approach (Equation 5). The solution is approximated numerically using `deSolve::ode()`. By default, the function uses the default settings (algorithm, tolerances, ...) of `ode()`, although the user can define other controls using the `...` argument.

As well as for predictions based on primary models only, this mode of calculation estimates the solution at the discrete time points passed through the argument `times`. Additionally, it requires the definition of the variation of the environmental conditions through the experiment (argument `env_conditions`). This must be defined as a `data.frame` (or `tibble`) with one column providing the elapsed time and as many columns as needed defining the values of the environmental conditions. For values of `time` not included in the data, the function calculates the values of the environmental conditions by linear interpolation.

The argument `secondary_models` serves to map a secondary model equation to each environmental factor using a nested list. Due to the use of the gamma approach, `predict_growth()` does not impose any limit to the number of environmental conditions. Nonetheless, every environmental factor must be defined at the same values of time.

Although models including secondary models are designed for describing the growth under dynamic environmental conditions, this modeling approach can also predict growth under static conditions. For that, one has to define a table with constant values for every environmental factor. The advantages of this approach are discussed in a dedicated package vignette.

As well as for predictions for `environment = "constant"`, the function returns an instance of '`GrowthPrediction`' with several S3 methods to ease the analysis of the results.

Fitting both primary and secondary models under dynamic environmental conditions

Passing `environment = "dynamic"` to the `fit_growth()` function allows the estimation of the parameters of both primary and secondary growth models from a dataset. In this case, the function requires two types of data. The first one, `fit_data` is the variation in the logarithm of the population size through the experiment. The second one, `env_conditions`, provides the variation of the environmental conditions. Both inputs are defined as a `tibble` (or `data.frame`) following the same conventions as the ones defined above for making predictions.

The fitting can be done using two different algorithms, according to the value of the argument `algorithm`: non-linear regression (using `FME::modFit()`), or an Adaptive Monte Carlo algorithm (Haario, Laine, Mira, and Saksman 2006, using `FME::modMCMC()`). Both fitting algorithms require the definition of initial guesses for every model parameter to estimate from the data. These are defined as a named numeric vector using the argument `start`. This can be defined based on information from the scientific literature or by a visual inspection of the predictions corresponding to a vector of parameter values. The function `check_growth_guess()` facilitates this second approach. Given a dataset and a list of model parameters, it generates a plot comparing the model prediction against the data.

Unlike for fitting when `environment = "constant"`, the package does not include any function to generate initial guesses of the model parameters based on some heuristic. The reason for this is the possible high parameter correlation for some experimental designs, where different parameter combinations results in similar predictions. For instance, under some conditions, the population growth could stop either because it has reached the stationary phase (defined by $\log N_{\max}$) or because the value of some environmental condition is above the one allowing growth (defined by $X_{\max}$). Consequently, in order to avoid pushing the fitting algorithm towards an arbitrary solution, a function based on heuristic rules is not implemented.

As already mentioned, this high parameter correlation can also cause poor parameter identifiability. In order to mitigate this, `fit_growth()` allows fixing any model parameter to

any value through the `known` argument. If the fitting was successful, the function returns an instance of '`GrowthFit`' with the same S3 methods as those implemented for models based on algebraic equations.

Note that `fit_growth()` does not implement any convergence check for models fitted by a Monte Carlo (MC) algorithm. Therefore, the convergence of the Markov chain has to be evaluated independently. This needs to be done by inspecting the instance of '`modMCMC`' that is included in `.$fit_results`. For details on how to evaluate the convergence of the algorithm, the reader is referred to the documentation of **FME** and references therein (Soetaert and Petzoldt 2010).

Fitting both primary and secondary models from several experiments (global fitting)

The `fit_growth()` function can also fit a unique growth model (based on primary and secondary model equations) to the results of several experiments obtained under different (static or dynamic) environmental conditions. This approach, sometimes called "global fitting" in predictive microbiology, is triggered by passing `approach = "global"`. This method is also recommended when the experimental method has low precision, potentially causing relevant changes in the environmental conditions between independent repetitions.

In this case, the input must be formatted in a similar way as when `approach = "single"`. It also requires two types of information: the variation in the population size through the experiment and the experimental conditions during the experiment. However, because the model is fitted to several experiments, the arguments `fit_data` and `env_conditions` must be defined as (preferably named) lists, where each entry defines the observations or environmental conditions as a `tibble` (or `data.frame`) following the same conventions as described above.

As well as for the standard approach, the model can be fitted either using regression or an Adaptive Monte Carlo algorithm, according to the value of the argument `algorithm`. In order to facilitate the definition of initial guesses for the model parameters, `check_growth_guess()` can compare the predictions corresponding to a model with the experimental data.

Global fitting is affected by similar issues related to poor parameter identifiability as described above for growth models fitted to data gathered under dynamic environmental conditions. Consequently, **biogrowth** does not include a function to automatically generate initial guesses based on heuristic rules. Furthermore, in order to mitigate potential parameter identifiability issues, any model can be fixed to any value using the `known` argument.

The function returns an instance of '`GlobalGrowthFit`', which implements similar S3 methods to facilitate the visualization and inspection of the model as those defined for '`GrowthFit`'.

Predictions including parameter uncertainty

Although the function `predict_growth()` and the `predict()` S3 methods can be used to make growth predictions for dynamic environmental conditions, they make discrete predictions without accounting for parameter uncertainty. For this type of calculation, the package implements `predictMCMC()`, which applies forward uncertainty propagation based on Monte Carlo simulations.

This function needs a model fitted using a Monte Carlo algorithm (passed through argument `model`). It generates a sample of model parameters of size `niter` by sampling with replacement the draws from the Markov chain. Then, for each sampled parameter vector, it calculates the

Quantity	Identifier	Description
$\log_{10} N$	`logN`	Logarithm of the population size
$\log_{10} N_0$	`logN0`	Logarithm of the initial population size
N	`N`	Population size
N_0	`N0`	Initial population size
t	`t`	Elapsed time
C	`C`	Log-increase in population size ($\log_{10} N_{\max} - \log_{10} N_0$)
μ	`mu`	Maximum growth rate (by default in log10 scale)
λ	`lambda`	Duration of the lag phase
ν	`nu`	Sharpness of the transition between growth phases in the Richards model
Q	`Q`	Variable of the Baranyi model describing the lag phase
Q_0	`Q0`	Initial value of Q
$X_{\min}$	`xmin`	Minimum value of X enabling growth
X_{opt}	`xopt`	Optimum value of X for population growth
$X_{\max}$	`xmax`	Maximum value of X enabling growth
n	`n`	Order of the secondary model
μ_{opt}	`mu_opt`	Value of μ under optimal environmental conditions for growth

Table 2: Summary of model parameters in **biogrowth**.

growth curve at the discrete time points defined in `times` by calling `predict_growth()`. The calculation is done for the environmental conditions defined as a `tibble` (or `data.frame`) in the `env_conditions` argument, following the conventions described above. Note that this input must have the same environmental conditions as those used for model fitting.

The function uses the distribution of the Monte Carlo simulations as an estimate for the distribution of the population size. The precision of this estimate depends both on the number of iterations and the precision of the posterior distribution estimated by the adaptive Monte Carlo algorithm. For this reason, it is essential when using `predictMCMC()` to also analyze in detail the convergence of the Markov Chain.

The function returns an instance of '`MCMCgrowth`'. This subclass of '`list`' provides direct access to the results of the calculations. It also implements S3 methods for `print()` and `plot()` to facilitate the analysis of the results of the simulations.

3.5. Mapping equations and parameters in biogrowth

The functions in the **biogrowth** package use character keys to identify model equations and map them to environmental factors (for secondary models). These can be retrieved using `primary_model_data()` and `secondary_model_data()`. When called without any argument, these functions returns a character vector with the keys corresponding to each primary or secondary growth model included in the package respectively.

The functions in the package follow a similar approach for mapping model parameters (e.g., for making predictions or to define initial values for model fitting). Passing a valid key to `primary_model_data()` or `secondary_model_data()` returns a list with meta-information on the model. This includes the entry `pars`, which is a character vector with the identifiers

for the selected model (summarized in Table 2). The output of these functions also includes `identifier` (the character key identifying the model), `name` (the whole name of the model), `model` (function with the model equation) and `ref` (reference to the scientific article where the model was defined). This meta-information is used by the functions in **biogrowth** to make several consistency check of the models before doing any calculation. These tests can be disabled by passing `check = FALSE` to the relevant function.

The functions `fit_growth()` and `predict_growth()` use slightly different approaches for mapping the parameters of secondary models. In `fit_growth()`, each secondary model is defined as a named list. This list has an entry named `model` defining the model equation. Then, the values of the model parameters are defined as additional entries whose names match those defined in `secondary_model_data()`. Once each secondary model has been defined, they are mapped to the environmental conditions by gathering them in a single list. In this list, each name must match the column names in `env_conditions`.

On the other hand, `fit_growth()` uses named numeric vectors to define model parameters. Then, each parameter of the secondary model must be defined as *factor name* + *_* + *parameter key* (as per `secondary_model_data()`). For instance, the parameter $X_{\min}$ for the environmental factor "temperature" would be defined as `temperature_xmin`. Then, the model equations are defined using the `model_keys` argument. The illustrations below show a practical illustration of both procedures. The package documentation shows additional examples.

By default, all the calculations in **biogrowth** are done assuming that both $\log N$ and μ are defined in log10 base. Nonetheless, this can be changed by passing different values to `logbase_logN` and `logbase_mu` of the relevant functions.

3.6. Statistical methods for model selection and comparison

The functions in **biogrowth** give the flexibility to fit different type of models. This refers both to using different model equations and to fixing any number of model parameters. These models can be compared using `compare_growth_fits()` and `compare_secondary_fits()`. Both functions take as argument a list of models fitted using the functions implemented in the package. It is advised that this list is named, as these names are kept for reporting the results.

Model selection is performed through several statistical indexes. The main one is the Akaike information criterion (AIC). The package includes the finite sample size correction in the calculation of AIC, defining it as

$$\text{AIC} = 2 \cdot k - 2 \cdot \log L[x] + \frac{2 \cdot k^2 + 2 \cdot k}{n - k - 1}$$

where k is the number of model parameters and n is the number of data points. The term $\log L[x]$ refers to the log-likelihood of the data. In **biogrowth**, it is calculated based on the probability density of a normal distribution with expected value given by the prediction of the fitted model and the standard deviation by the standard error of the residuals.

Apart from that, `compare_growth_fits()` and `compare_secondary_fits()` return the number of degrees of freedom, calculated as the number of rows in the data minus the number of estimated model parameters ($df = n - k$). Therefore, the fixed model parameters are not considered in this calculation. It also reports the mean error (ME) and root mean squared

error (RMSE)

$$\text{ME} = \frac{1}{n}\sum_i e_i$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_i e_i^2}$$

where e_i is the residual corresponding to observation i and n is the number of data points.

The analysis of the results of these functions is done by examining the objects they return (an instance of either '`GrowthComparison`' or '`SecondaryComparison`'). Both implement an S3 method for `print()` that provides a short summary of the models analyzed, as well as a table of the models arranged by AIC. They also implement an S3 method for `plot()` that can be used to compare the models visually. This includes comparing the fitted curves, as well as the model parameters. They also implement methods for `coef()` and `summary()` to retrieve the statistical indexes used for model comparison and selection.

3.7. Method to estimate the elapsed time to reach a given population size

The time to reach a given population size is often a quantity of great interest in studies related to population growth. In **biogrowth**, this can be estimated using `time_to_size()`. The function has two mandatory arguments. The first one (`model`) is an instance of '`GrowthPrediction`', '`GrowthFit`', '`GlobalGrowthFit`', '`GrowthUncertainty`' or '`MCMCgrowth`' with the growth model. Therefore, `time_to_size()` can be used both for models parameters defined manually or estimated from data. The second mandatory argument, `size`, defines the target population size in log units. By default, `time_to_size()` assumes that `size` is defined in the same units as those used in `model`. Nonetheless, this can be changed with the `logbase_logN` argument.

By default, `time_to_size()` estimates the time by linear interpolation using `approx()` from base R, setting the population size as the explanatory variable (`x`) and the elapsed time as response variable (`y`). Note that the precision of the estimate depends strongly on the number of time points used for the growth predictions. Therefore, it is advisable to plot the growth curve before calling `time_to_size()` (e.g. using the `plot()` methods). If the curvature of the growth curve is not well described in the vicinity of the target population size, the prediction should be repeated increasing the number of time points. If the target population size is outside the range of the simulations, `time_to_size()` returns `NA`.

On the other hand, if `type = "distribution"`, the function accounts for parameter uncertainty in the calculations. As of **biogrowth** 1.0.3, this type of calculation is only compatible when the growth model is an instance of '`GrowthUncertainty`' or '`MCMCgrowth`'. The distribution is calculated by calling `time_to_size(type = "discrete")` for the growth curve corresponding to each Monte Carlo simulation of the `model`. Therefore, it generates as many values of time as iterations in the original model, which serves as an estimate for the distribution of the time to reach the target population accounting for parameter uncertainty. In this case, the function returns an instance of '`TimeDistribution`' with the results of the simulations that implements S3 methods for `print()`, `summary()` and `plot()` to ease the analysis of the results.

Note that the precision of these calculations depend on several simulation settings that must be checked by the user. First of all, the elapsed time is calculated several times with `type = "discrete"`. Hence, it has the same dependency on the number of time points. Moreover,

it also depends on the size of the parameter sample. It is thus advisable to repeat the simulations several times to evaluate the stability of the solution before defining a seed (using `set.seed()`). Finally, care must be taken when the target population size is close to the initial or maximum population size because it is possible that some of the Monte Carlo simulations did not reach the target count. In that case, the calculation would return `NA` for some Monte Carlo samples. Because the quantiles are calculated omitting `NA`s, the summary table generated by `time_to_size()` would be biased. Therefore, it is advisable to carefully evaluate the results of `predict_growth_uncertainty()` before using this function.

4. Illustrations

4.1. Growth models from published parameters

One way to build growth models is by using growth parameters reported in scientific publications (often called "forward problem"). This approach is illustrated here using a published growth model for *Bacillus cereus* (Carlin, Albagnac, Rida, Guinebretière, Couvert, and Nguyen-the 2013). This article used a Baranyi primary growth model and a cardinal parameter model to describe the effect of pH, temperature and water activity in the growth kinetics of several phylogenetic groups of *B. cereus*. For details on the methodology, the reader is referred to the original article (Carlin *et al.* 2013).

In this illustration, we will simulate the growth of this microorganism in a domestic refrigerator using data already published (Jofré, Latorre-Moratalla, Garriga, and Bover-Cid 2019). The raw data was kindly provided by the original authors of the manuscript by personal communication. This dataset includes measurements of the temperature in several domestic refrigerators in Catalonia (Spain). For further details about the methodology, the reader is referred to the original article by the authors. The is included in the package under the name `refrigeratorSpain`.

```
R> data("refrigeratorSpain", package = "biogrowth")
R> head(refrigeratorSpain)

# A tibble: 6 x 3
   time    A1    A2
  <dbl> <dbl> <dbl>
1 0       5.5   7
2 0.167   6.6   7
3 0.333   7.3   7
4 0.5     6.8   6.9
5 0.667   6     6.8
6 0.833   6.1   6.9
```

In order to account for the effect of temperature on μ, we need to define a secondary growth model. Although the original authors reported growth models for several phylogenetic groups of *B. cereus*, in this illustration we only use the one for strain RIVM BC120 ($T_{\text{opt}} = 36.8°\text{C}$, $T_{\text{min}} = 1.4°\text{C}$, $T_{\text{max}} = 41.0°\text{C}$ for a CPM model of order 2). Secondary models for `predict_growth()` are defined as a named list. They must contain an entry named `model`

with a valid key, which can be retrieved by calling `secondary_model_data()` without any argument.

```
R> secondary_model_data()
```

```
[1] "CPM"           "Zwietering"    "fullRatkowsky"
```

In this case, we will use a "CPM" model (the model used by the original authors). The values of the model parameters are defined using the same list, identifying them using valid keys that can be retrieved passing a model key to `secondary_model_data()`

```
R> meta_info <- secondary_model_data("CPM")
R> meta_info$pars
```

```
[1] "xmin" "xopt" "xmax" "n"
```

Therefore, the secondary model with $T_{\text{opt}} = 36.8°\text{C}$, $T_{\text{min}} = 1.4°\text{C}$, $T_{\text{max}} = 41.0°\text{C}$, $n = 2$ would be defined as

```
R> sec_temperature <- list(model = "CPM", xmin = 1.4,
+    xopt = 36.8, xmax = 41.0, n = 2)
```

The next step is mapping this secondary model to a column in the data. First, we will make the calculation for refrigerator "A1". Therefore, we need to map the temperature in the model to column `A1` in the data using a named list. Note that `fit_growth()` uses the gamma approach, so it can account for any number of environmental factors. If we wanted to account for additional environmental factors, we would include additional entries in this list (see the package vignettes for examples).

```
R> my_secondary <- list(A1 = sec_temperature)
```

Then, we need to define the parameters of the primary model. Namely, the values of μ_{opt}, N_{max}, N_0 and Q_0. The value of μ_{opt} was reported as 2.61 ln CFU/h by (Carlin *et al.* 2013). For illustration purposes, we will define an initial population of 100 CFU/g, whereas we will define an N_{max} of 10,000,000 CFU/g, a typical value for *B. cereus*. As described above, parameter Q_0 defines the duration of the lag phase. We will assign a large value to this parameter (1,000) to have a model without a lag phase.

```
R> my_primary <- list(mu_opt = 2.61, Nmax = 1e7, N0 = 100, Q0 = 1e3)
```

Before calling `predict_growth()`, we just need to define the time points where the solution is calculated. We will define 1,000 uniformly distributed points between 0 and the maximum time in the data. Then, we can pass these arguments to `predict_growth()`. Note that, by default, the calculations are done in base 10. Because μ_{opt} was reported in base e by the original authors, we need to pass the argument `logbase_mu = exp(1)`.

```
R> my_times <- seq(0, max(refrigeratorSpain$time), length = 1000)
R> cereus_A1 <- predict_growth(environment = "dynamic",
+    my_times, my_primary, my_secondary, refrigeratorSpain,
+    logbase_mu = exp(1)
+  )
```

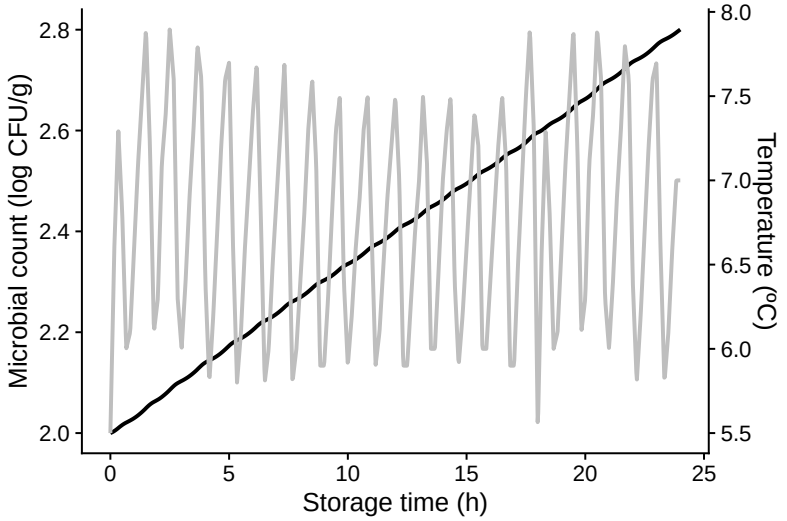

Figure 3: Prediction for the growth of *B. cereus* in a domestic refrigerator (black line). The grey line is the temperature profile.

The instance of '`GrowthPrediction`' returned by the function implements several S3 methods to facilitate the analysis. The `plot()` method allows a visualization of the growth curve.

```
R> plot(cereus_A1, add_factor = "A1", label_y2 = "Temperature (ºC)",
+    label_y1 = "Microbial count (log CFU/g)", label_x = "Storage time (h)",
+    line_type2 = "solid", line_col2 = "grey")
```

In order to make predictions for the other refrigerator, we can just change the mapping in the secondary model and call again `predict_growth()`.

```
R> my_secondary <- list(A2 = sec_temperature)
R> cereus_A2 <- predict_growth(environment = "dynamic",
+    my_times, my_primary, my_secondary,
+    refrigeratorSpain, logbase_mu = exp(1)
+  )
```

The fact that the `plot()` method returns an instance of `ggplot` has several advantages. For instance, we can pass them to `cowplot::plot_grid()` to compare the model predictions under the three conditions, to evaluate whether the frequency of the temperature oscillations have a relevant effect in the model predictions.

```
R> plot_grid(labels = c("A1", "A2"), scale = .9,
+    plot(cereus_A1, add_factor = "A1", ylims = c(2, 3),
+      label_y2 = "Temperature (ºC)",
+      label_y1 = "Microbial count (log CFU/g)",
+      label_x = "Storage time (h)",
+      line_type2 = "solid", line_col2 = "grey"
+    ),
```

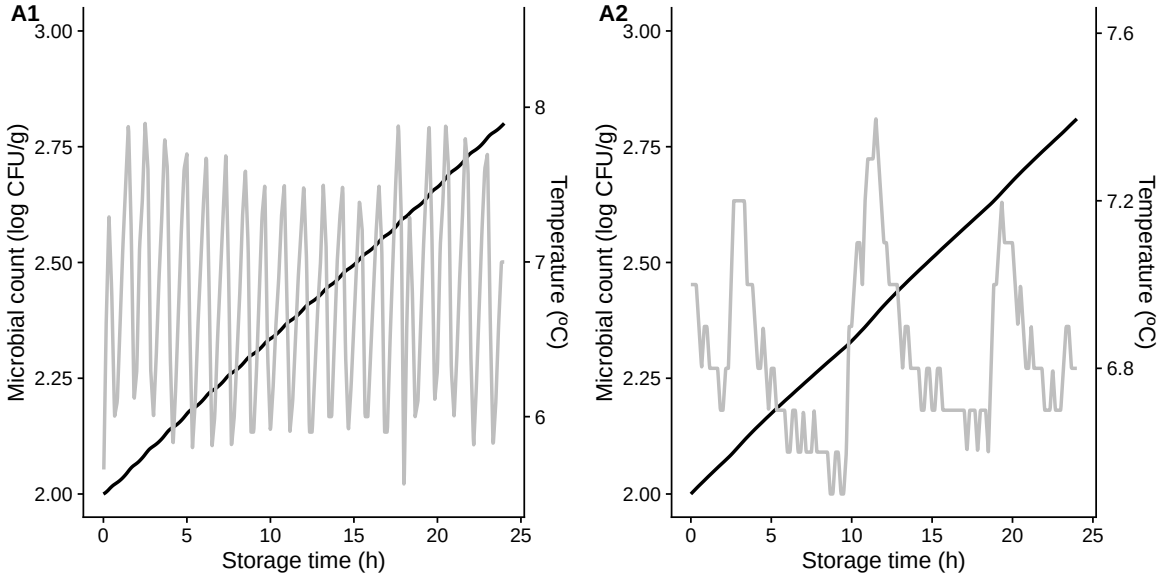

Figure 4: Prediction for the growth of *B. cereus* in two domestic refrigerators (black line). The grey line is the temperature profile.

```
+    plot(cereus_A2, add_factor = "A2", ylims = c(2, 3),
+      label_y2 = "Temperature (ºC)",
+      label_y1 = "Microbial count (log CFU/g)",
+      label_x = "Storage time (h)",
+      line_type2 = "solid", line_col2 = "grey")
+  )
```

```
Warning: Removed 1 row(s) containing missing values (geom_path).
```

4.2. Growth models by fitting primary models to growth data

Although the methodology for modeling population growth included in **biogrowth** was developed within the field of predictive microbiology, similar models are commonly used in other fields. This is illustrated here, where we model the increase in the number of tractors registered in Greece in a 45 year period. The data was obtained from FAOSTAT (http://www.fao.org/faostat/en/#data/RM) filtering by country selecting "Greece", by element selecting "In Use", by item selecting "Agricultural tractors", and by year selecting every year available. A different subset of this dataset was already analyzed using the Gompertz and Logistic models in a previous study (Nguimkeu 2014). The data is included in the package under the name `greek_tractors`.

The data was downloaded and saved as a CSV file (comma-separated values) that is included as supplementary material to this article. It can be loaded using `readr::read_csv()`.

```
R> data("greek_tractors", package = "biogrowth")
```

We will apply two transformations to the data before fitting the models. First, because the models implemented in **biogrowth** usually describe the logarithm of the population size, the

number of tractors is log-transformed. Second, to ease model interpretation (especially for the lag phase duration), we shift the observations, making the first year in the data (1961) year "0".

```
R> greek_tractors <- greek_tractors %>%
+    mutate(logtractors = log10(Value),
+      t_model = Year - min(Year)
+    )
```

We will use the `fit_growth()` function to estimate the model parameters of a logistic growth model. As already mentioned, models are defined in **biogrowth** using a character key, which can be retrieved calling `primary_model_data()` without any arguments.

```
R> primary_model_data()

[1] "modGompertz" "Baranyi"     "Trilinear"   "Logistic"    "Richards"
```

Passing a valid model key to `primary_model_data()` returns a list with meta-data of the model, including the identifiers of each model parameter:

```
R> meta_info <- primary_model_data("Logistic")
R> meta_info$pars

[1] "logN0"  "C"      "mu"     "lambda"
```

The fitting algorithms included in this function require the definition of initial guesses for every model parameter fitted from the data ($\log N_0$, C, μ and λ). Although this can be done by visual inspection or from historical data, the `make_guess_primary()` function can be used to obtain initial guesses automatically. Note that we are passing the `formula` argument to indicate the column names for the population size and the elapsed time.

```
R> my_guess <- make_guess_primary(greek_tractors,
+    primary_model = "Logistic", formula = logtractors ~ t_model)
R> my_guess

     logN0         mu     lambda          C
4.35468455 0.02717687 5.00000000 1.05989775
```

We can visually evaluate whether this guess is reasonable using `check_growth_guess()`.

```
R> check_growth_guess(greek_tractors,
+    model_keys =  list(primary = "Logistic"), guess = my_guess,
+    formula = logtractors ~ t_model
+  )
```

The plot shows that the line is reasonably close for an initial guess to be used for the fitting algorithm. Therefore, we can call `fit_growth()` to get the parameter estimates.

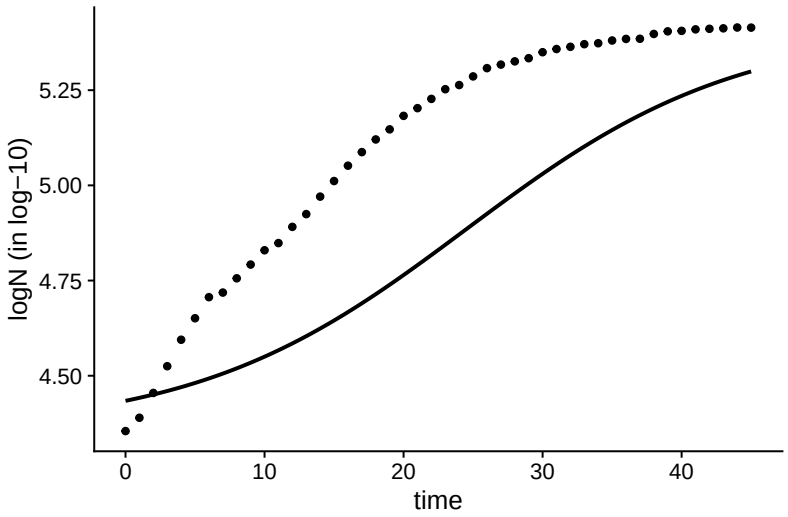

Figure 5: Visualization of the initial guess generated automatically for the data on the number of tractors.

```
R> greek_logistics <- fit_growth(greek_tractors,
+    list(primary = "Logistic"), start = my_guess,
+    known = c(), formula = logtractors ~ t_model
+  )
R> greek_logistics

Primary growth model fitted to data

Growth model: Logistic

Estimated parameters:
       logN0           mu       lambda            C
  3.60457777   0.04781977 -16.04746607   1.84390683

Fixed parameters:
NULL

Parameter mu defined in log-10 scale
Population size defined in log-10 scale
```

We can use the S3 `plot` method included in the package to visually evaluate that the model correctly describes the data. From the figure, we can see that the function was able to fit the model to the data.

```
R> plot(greek_logistics,
+    label_x = "Year", label_y1 = "log10 of the number of tractors")
```

Then, we can use the `summary()` method to retrieve the values of the parameter estimates, as well as their standard errors.

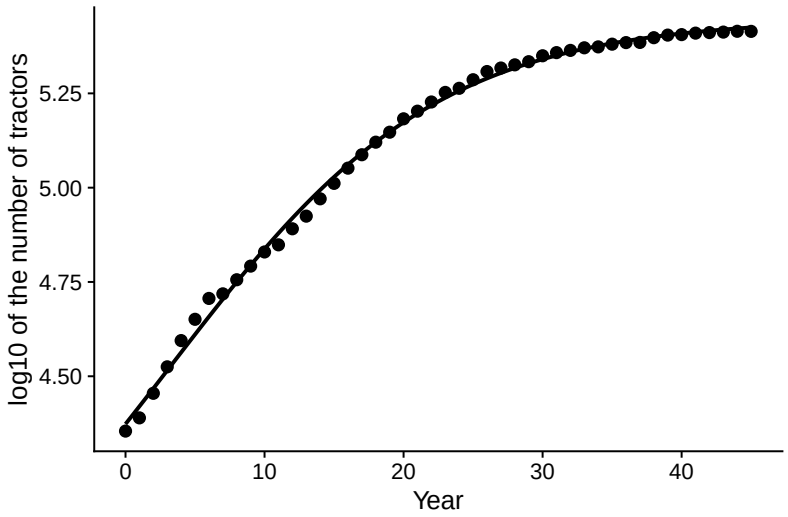

Figure 6: Fit of the logistic model to the number of tractors in Greece between 1961 (year 0 in the plot) and 2006.

```
R> summary(greek_logistics)

Parameters:
         Estimate Std. Error t value Pr(>|t|)
logN0    3.604578   0.165268  21.810  < 2e-16 ***
mu       0.047820   0.001897  25.212  < 2e-16 ***
lambda -16.047466   2.787844  -5.756 8.90e-07 ***
C        1.843907   0.172679  10.678 1.53e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0171 on 42 degrees of freedom

Parameter correlation:
         logN0      mu  lambda       C
logN0   1.0000 -0.9069  0.9892 -0.9992
mu     -0.9069  1.0000 -0.8370  0.8947
lambda  0.9892 -0.8370  1.0000 -0.9921
C      -0.9992  0.8947 -0.9921  1.0000
```

It could seem remarkable that a negative value is estimated for parameter λ, which quantifies the duration of the lag phase. In a different context, this could indicate a deficiency of the model. However, agricultural tractors have been available since the beginning of the XXth century, whereas the earliest observation in the data is from 1961. Therefore, in this case, it is more appropriate to use the mathematical interpretation of parameter λ, defined as the value of x, where a line with slope μ tangent to the point with maximum growth rate cuts a horizontal line with y intercept $\log N_0$. Then, a negative value for λ implies that the onset of exponential growth for the population took place approximately in 1945 ($1961 - 16 = 1945$).

To better illustrate this, we will also fit a growth model fixing parameter λ to zero using the `known` argument.

```
R> greek_logistics_noLag <- fit_growth(greek_tractors,
+    list(primary = "Logistic"), start = c(mu = 0.03, logN0 = 4, C = 1),
+    known = c(lambda = 0), formula = logtractors ~ t_model
+  )
```

Although the fit of the new model could also be analyzed using the S3 methods for plotting or statistical summary defined for a growth fit, the function `compare_growth_fits()` includes several functions for model comparison/selection. It takes as only input a list of model fits and returns an instance of '`GrowthComparison`' with several S3 methods for model comparison and selection. In this sense, the `print()` method lists all the models arranged by Akaike information criterion.

```
R> tractor_comparison <- compare_growth_fits(
+    list(`Logistic`= greek_logistics,
+      `Logistic - no lag` = greek_logistics_noLag
+    )
+  )
R> tractor_comparison

Comparison between models fitted to data under isothermal conditions

Statistical indexes arranged by AIC:

# A tibble: 2 x 5
  model               AIC    df            ME   RMSE
  <chr>             <dbl> <int>         <dbl>  <dbl>
1 Logistic          -239.    42 -0.000000156  0.0163
2 Logistic - no lag -179.    43 -0.0000000655 0.0321
```

The results show that the "Logistic" model (i.e., the one with the negative value of λ) would be preferred according to the AIC. Note that the models are named according to the names defined in the list passed to `compare_growth_fits()`. For this reason, it is recommended to name this list.

Besides the comparison by statistical indexes, `GrowthComparison` also includes a `plot()` method to visually compare the model fits.

```
R> plot(tractor_comparison)
```

This plot shows that the model where we fixed $\lambda = 0$ clearly deviates with respect to the observations at the beginning of the time series. This is further emphasized, when `type = 3` is passed to the plotting method to generate a plot of the residuals.

```
R> plot(tractor_comparison, type = 3)

`geom_smooth()` using formula 'y ~ x'
```

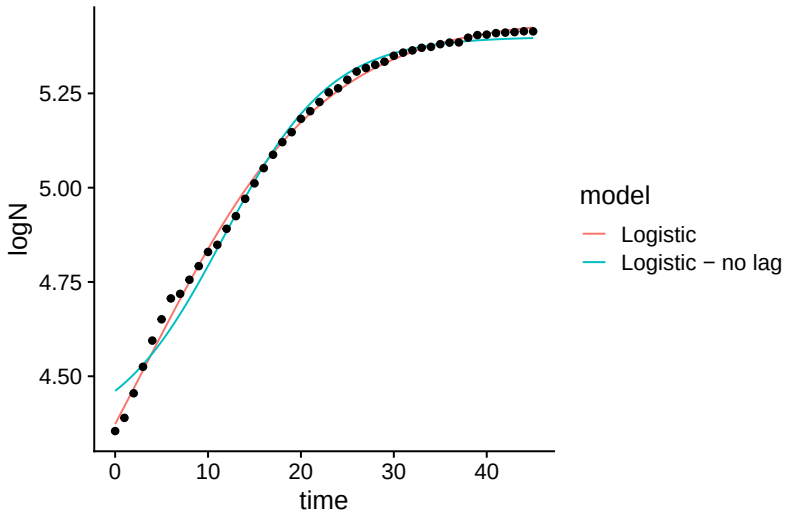

Figure 7: Comparison of the model fits of the two logistic models to the tractor data

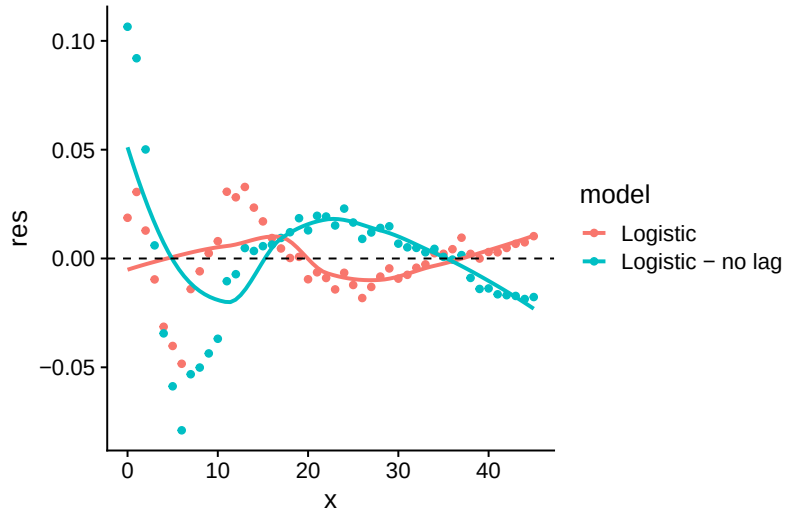

Figure 8: Comparison of the residuals of the two logistic models fitted to the tractor data.

This figure shows that the residuals are larger at the beginning of the experiment for the *Logistic – no lag* model, further confirming the conclusions drawn from the previous figure. The effect of this deviation on the parameter estimates can be observed by passing `type = 2` to the `plot()` method.

```
R> plot(tractor_comparison, type = 2)
```

This plot shows that, although both models estimate similar values for μ, the model without lag estimates a larger value for $\log N_0$ and smaller for C. This result can be visualized using the `predict()` method to calculate the model predictions for time points on a broader time range than the ones included in the data. We will calculate the predictions in the time range

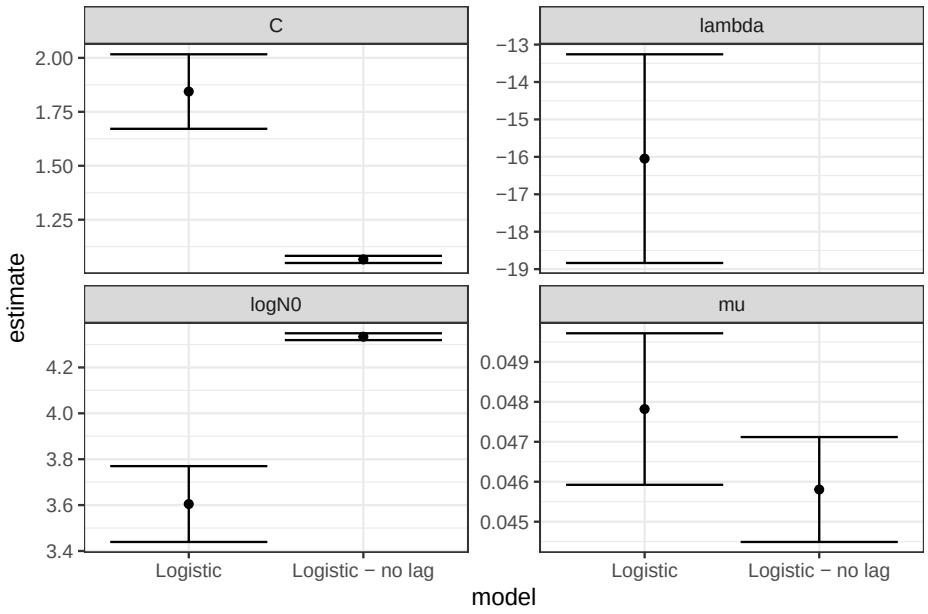

Figure 9: Comparison of the model parameters estimated with both logistic models from the tractor data. Dots represent estimated values and the error bars their standard errors.

between 1950 and 2010, so the model will be extrapolated both forwards and backwards in time.

```
R> library("ggplot2")
R> tibble(
+    year = 1950:2020,
+    t_model = year - min(greek_tractors$Year)
+    ) %>%
+    mutate(Logistic = predict(greek_logistics, times = t_model),
+      `Logistic no lag` = predict(greek_logistics_noLag, times = t_model)
+    ) %>%
+    pivot_longer(-c(t_model, year),
+      names_to = "model", values_to = "logtractors") %>%
+    ggplot() +
+    geom_line(aes(x = year, y = logtractors, colour = model), size = 1) +
+    geom_point(aes(x = Year, y = logtractors),
+      data = greek_tractors, size = 2) +
+    ylab("log10 of the number of tractors") + xlab("Year") +
+    theme_cowplot() +
+    theme(legend.title = element_blank(), legend.position = "top")
```

This plot shows that, by fixing $\lambda = 0$, the model estimates that the number of tractors in Greece before 1961 is practically equal to the number observed at that time point. On the other hand, the model that fits every parameter predicts that the number of tractors in the country increased exponentially since 1950. Considering that tractors were available since the early XXth century, and that there was economic growth in Greece between 1950 and 1960,

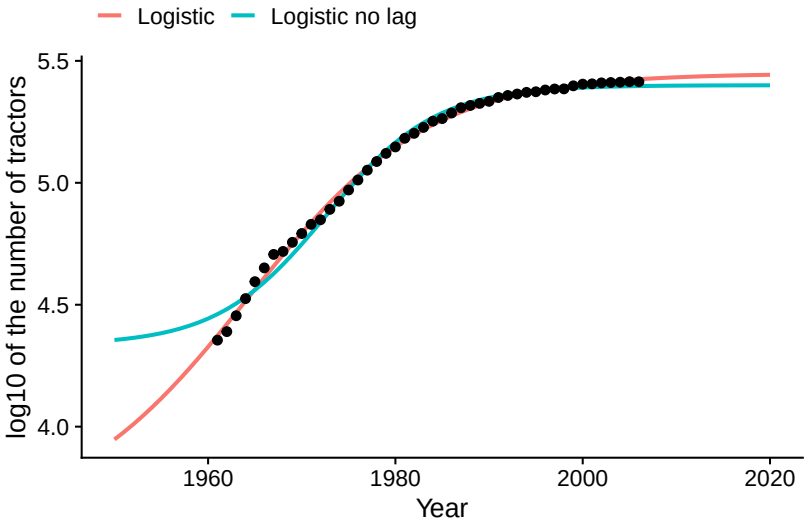

Figure 10: Comparison between the fit of both models (including extrapolation) to the number of tractors in Greece between 1961 and 2006 when the lag phase is fixed to one.

the prediction of the model fitting every model parameter seems more reasonable. Together with the fact the model without a lag phase deviates from the trend of the data points at the beginning of the experiment, it is reasonable to conclude that this dataset is better described by the model that considers a lag phase, even if the value of λ is negative.

The function `predict_growth_uncertainty()` enables including parameter uncertainty in the model predictions. It is thus intended for cases where variation is a relevant part of the problem studied, either because the system is highly uncertain or because variability is an inherent part of it (e.g. in microbial risk assessment, EFSA Scientific Committee *et al.* 2018). Although these distributions can be defined by hand, they can also be easily generated from an instance of '`GrowthComparison`' using the `coef()` method.

```
R> pars <- coef(tractor_comparison) %>%
+    filter(model == "Logistic") %>%
+    select(-model, par = parameter, mean = estimate, sd = std.err) %>%
+    mutate(scale = "original")
R> pars

# A tibble: 4 x 4
  par        mean      sd scale
  <chr>     <dbl>   <dbl> <chr>
1 logN0    3.60   0.165   original
2 mu       0.0478 0.00190 original
3 lambda -16.0    2.79    original
4 C        1.84   0.173   original
```

Then, we can call the `predict_growth_uncertainty()` to include the parameter uncertainty in the predictions. The instance of `GrowthUncertainty` has several methods to facilitate

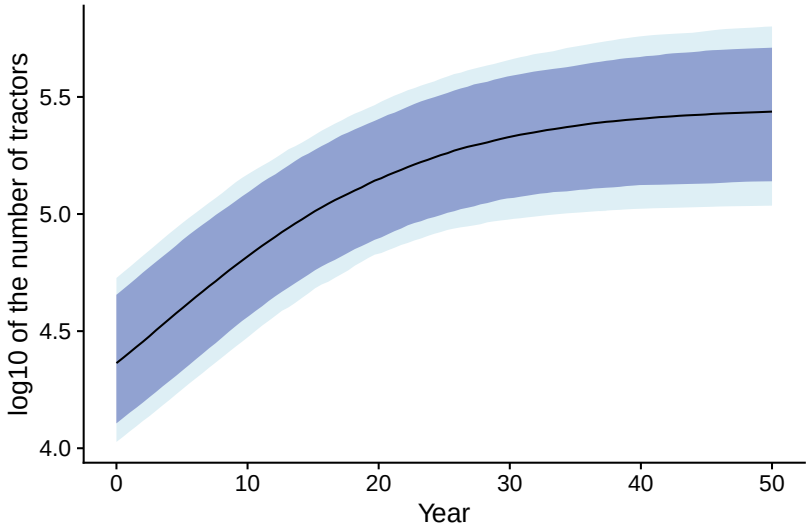

Figure 11: Prediction band for the number of tractors in Greece accounting for parameter uncertainty. The lightblue ribbon shows the interval for the 0.9 level and the darkblue the 0.8 interval.

the analysis. It includes a `plot()` method to visualize the prediction band, where the line represents the median of the simulations, and the shaded area the 5th, 10th, 90th and 95th quantiles.

```
R> set.seed(12412)
R> greek_uncertainty <- predict_growth_uncertainty(
+    greek_logistics$primary_model,
+    times = seq(0, 50, length = 100),
+    n_sims = 1500,
+    pars)
R> plot(greek_uncertainty,
+    ribbon80_fill = "darkblue", ribbon90_fill = "lightblue") +
+    xlab("Year") + ylab("log10 of the number of tractors")
```

A final aspect that is often of great interest when using growth models is the elapsed time required to reach a given population size. This can be estimated using the `time_to_size()` function, which can make either a discrete calculation or estimate a distribution for this variable. As an example, we will calculate the distribution of the time for a target size of $10^{4.7}$ (chosen arbitrarily for this illustration).

```
R> distrib_to_4p7 <- time_to_size(greek_uncertainty, size = 4.7,
+    type = "distribution")
R> distrib_to_4p7

Distribution of the time required to reach a target population size

# A tibble: 1 x 5
```

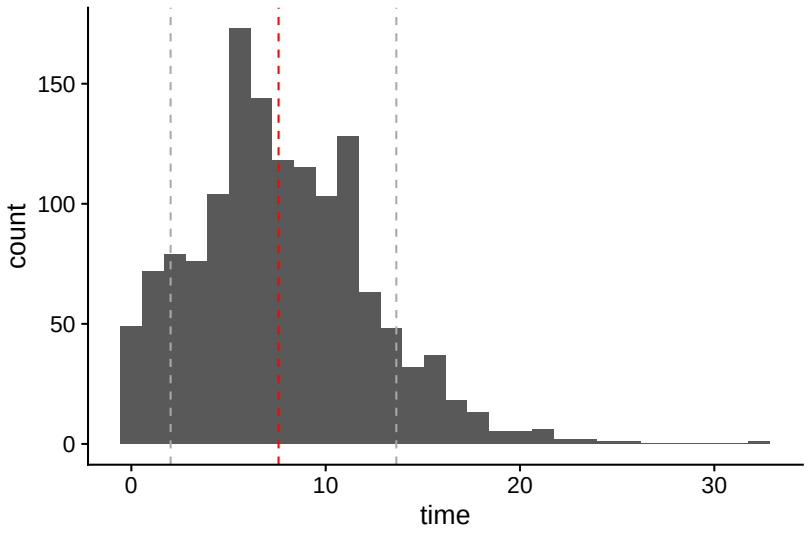

Figure 12: Distribution of the time to reach a population size of 4.7 log for the tractor data. The vertical, red line represents the median of the results. The vertical, gray lines the 10th and 90th percentiles.

```
  m_time sd_time med_time   q10   q90
   <dbl>   <dbl>    <dbl> <dbl> <dbl>
1   7.72    4.47     7.58  2.02  13.6
```

The instance of '`TimeDistribution`' includes several S3 methods to facilitate the analysis of the results. For instance, the `plot()` method shows a histogram of the solution, where the median is shown as a red dashed line and the 10th and 90th quantiles using gray dashed lines.

```
R> plot(distrib_to_4p7)

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 105 rows containing non-finite values (stat_bin).
```

4.3. Growth models by fitting secondary growth models

A different approach for building growth models is by first performing several growth experiment under constants environmental conditions. Then, a primary model is fitted for each experiment (e.g., following the method showed in the previous illustration). Finally, a secondary growth model can be fitted to this dataset to estimate the parameters of the secondary models according to the gamma approach. This approach is supported in **biogrowth** by `fit_secondary_growth()`. We illustrate the use of this function using the `example_cardinal` dataset, which contains simulated data. It is included within the package, so it can be loaded as `data("example_cardinal")`.

```
R> data("example_cardinal", package = "biogrowth")
R> head(example_cardinal)
```

```
  temperature pH           mu
1    0.000000  5 9.768505e-04
2    5.714286  5 2.624919e-03
3   11.428571  5 0.000000e+00
4   17.142857  5 1.530706e-04
5   22.857143  5 2.301817e-05
6   28.571429  5 3.895598e-04
```

The dataset includes simulated values of μ for several experiments at different values of pH and temperature. Each row of the dataset (64 in total) would represent one individual experiment. We need to assign a secondary model equations to each environmental factor. In this example, we will assign a cardinal parameter model (key `CPM`) to both factors.

```
R> sec_model_names <- c(temperature = "CPM", pH = "CPM")
```

The `fit_secondary_growth()` function uses non-linear regression for parameter estimation. This algorithm requires the definition of initial guesses for every model parameter to estimate from the data. Although they can be defined based on experience or visualizations, `make_guess_secondary()` can be used to obtain initial guesses.

```
R> my_guess <- make_guess_secondary(example_cardinal, sec_model_names)
R> print(my_guess)
```

```
         mu_opt temperature_xmin temperature_xopt temperature_xmax
       1.234784         0.000000        34.285714        40.000000
  temperature_n          pH_xmin          pH_xopt          pH_xmax
       2.000000         5.000000         6.428571         7.000000
           pH_n
       2.000000
```

As a first step, we will estimate every parameter in the model passing an empty vector to `known_pars`.

```
R> fit_cardinal <- fit_secondary_growth(example_cardinal, my_guess,
+    sec_model_names, known_pars = c())
```

```
Warning in sqrt(my_data$pred_mu): NaNs produced

Warning in sqrt(my_data$pred_mu): NaNs produced

Warning in nls.lm(par = Pars, fn = Fun, control = Contr, ...):
lmdif: info = -1. Number of iterations has reached `maxiter' == 100.
```

The fitting algorithm returns several warnings when doing the calculation. This is often an indication that the model fitted should be analyzed with special care. The instance of '`FitSecondaryGrowth`' returned by the function implements several S3 methods to help in this analysis. Namely, the `summary()` method provides the statistical information on the model fit.

```
R> summary(fit_cardinal)

Warning in summary.modFit(object$fit_results): Cannot estimate covariance;
system is singular

Parameters:
                 Estimate Std. Error t value Pr(>|t|)
mu_opt              1.026         NA      NA       NA
temperature_xmin    5.089         NA      NA       NA
temperature_xopt   34.855         NA      NA       NA
temperature_xmax   40.000         NA      NA       NA
temperature_n       1.140         NA      NA       NA
pH_xmin             5.138         NA      NA       NA
pH_xopt             6.476         NA      NA       NA
pH_xmax             7.000         NA      NA       NA
pH_n                3.010         NA      NA       NA

Residual standard error: 0.04741 on 55 degrees of freedom

Error in cov2cor(x$cov.unscaled): 'V' is not a square numeric matrix
```

In this case, the function returns an error and calculates `NA`'s for the standard errors of every model parameter. This is yet another indication of parameter identifiability issues. To mitigate this issue, `fit_secondary_growth()` gives the possibility to fix any model parameter (and is often necessary in this type of model fit). Often, the first candidates to fix in this type of model are the order of the models (n). In this case, we will fix the one for temperature to 1 and the one for pH to 2. We will also fix the $X_{\max}$ for temperature to 40°C, and $X_{\max}$ and $X_{\min}$ for pH to 7 and 5.2 respectively. Note that we also remove these parameters from `my_start` to indicate they should not be fitted.

```
R> known_pars <- list(
+    temperature_n = 1, temperature_xmax = 40,
+    pH_n = 2, pH_xmax = 7, pH_xmin = 5.2
+  )
R> my_start <- my_guess[c("temperature_xmin", "temperature_xopt",
+    "pH_xopt", "mu_opt")]
R> fixed_cardinal <- fit_secondary_growth(example_cardinal,
+    my_start, known_pars, sec_model_names)
```

Fixing these model parameters resolves some of the identifiability issues. This results in the regression algorithm being able to estimate standard errors for every model parameters.

```
R> summary(fixed_cardinal)

Parameters:
                 Estimate Std. Error t value Pr(>|t|)
temperature_xmin  6.11686    1.20602   5.072 4.06e-06 ***
```

```
temperature_xopt 35.69253    2.46344  14.489  < 2e-16 ***
pH_xopt           6.49849    0.02538 256.002  < 2e-16 ***
mu_opt            1.01051    0.09900  10.207 9.79e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05606 on 60 degrees of freedom

Parameter correlation:
                 temperature_xmin temperature_xopt    pH_xopt  mu_opt
temperature_xmin        1.000e+00       -6.296e-01  1.241e-08 -0.3830
temperature_xopt       -6.296e-01        1.000e+00 -2.909e-09  0.8884
pH_xopt                 1.241e-08       -2.909e-09  1.000e+00  0.1732
mu_opt                 -3.830e-01        8.884e-01  1.732e-01  1.0000
```

The instance of '`FitSecondaryGrowth`' also includes `plot()` methods to visualize the goodness of the model fit. By default it generates a scatter plot of observed against predicted values, including a gray line showing a linear regression model fitted to fits vs observations. This line can be compared against the black, dashed line showing a perfect fit (slope = 1, intercept = 0). In cases where there is a good agreement between both lines (as is the case here), one can conclude that the fitted model describes the general trend in the data.

```
R> plot(fixed_cardinal)
```

```
`geom_smooth()` using formula 'y ~ x'
```

The plot includes an alternative plotting method. Passing `which = 2`, shows a scatter plot individually for each environmental factor, showing both the model fits (black) and the observations (gray). Note that, because the gamma model considers several environmental factors, there are different model fits for each value of X (representing the effect of the other environmental factors). For this reason, in order to facilitate the comparison, passing `add_segment = TRUE` joins the observation and its corresponding fitted value. Moreover, passing `add_trend = TRUE` adds a trend line for both the fits and the observations by local regression.

```
R> plot(fixed_cardinal, which = 2, add_trend = TRUE, add_segment = TRUE)
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

Both plots show that the model describe adequately the data. Nonetheless, there are some conditions for which the model fits a value of $\mu = 0$, whereas the experiments observed some growth. Considering that the values of $X_{\max}$ for both pH and temperature were fixed, this could indicate that they were fixed to too narrow values. Also, note that the trend line shown in the second plot is not a model fit. The reason for this is that the gamma approach can have an arbitrary number of environmental condition, whereas each facet can only have one environmental factor. Therefore, this plot is only intended to aid in evaluating the goodness of the fit. For making model predictions, the `predict()` method should be used.

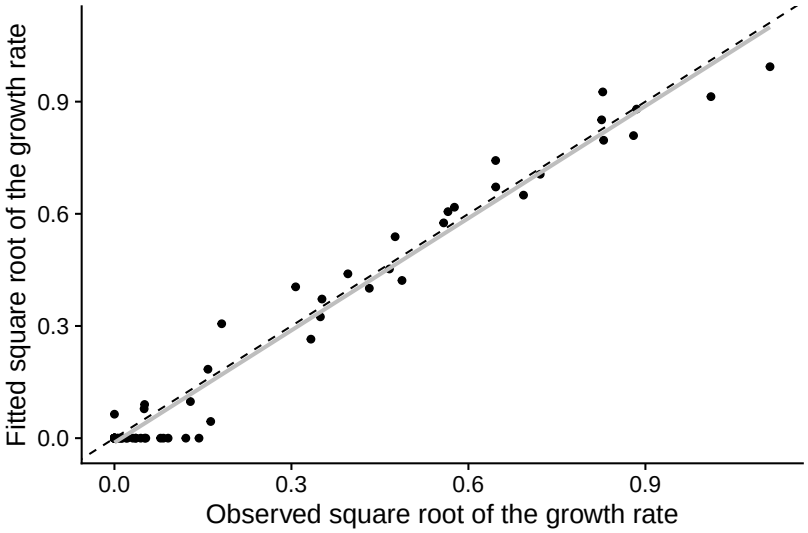

Figure 13: Comparison between the model fits and the predictions for the secondary model fitted to simulated data. The solid grey line is a linear regression model for fitted vs. observed and the dashed black line represents a perfect fit.

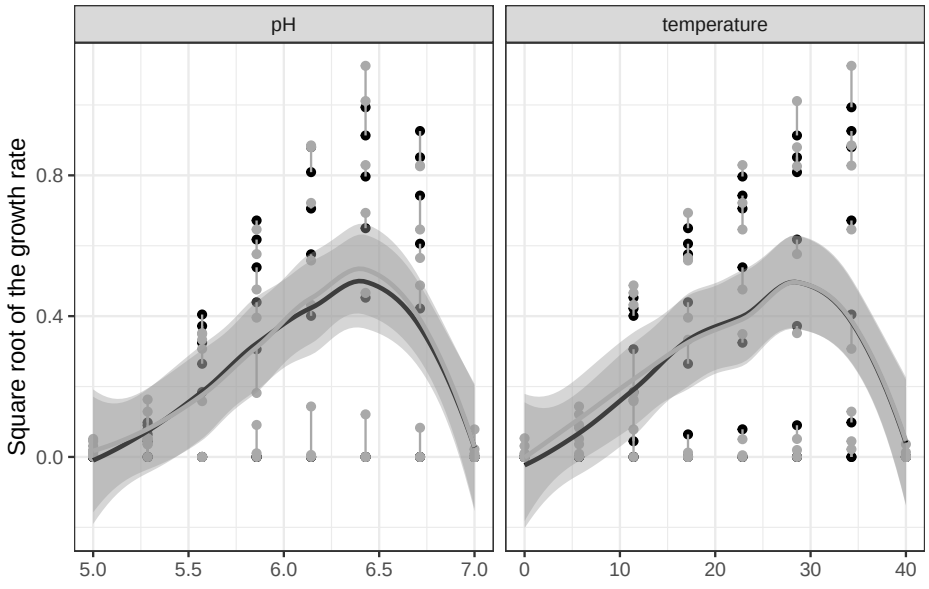

Figure 14: Comparison between the trend of the fitted model and the one of the observations for the secondary model fitted to simulated data. Observations are shown in gray and model fits in black.

4.4. Growth models for data under dynamic environmental conditions

This case study illustrates how to build growth models (primary and secondary models) using data gathered under dynamic environmental conditions. For this, we use a dataset gathered from the online database **ComBase** (**ComBase** ID: Lm_MpDyn_T1_R1) that describes the growth of *Listeria monocytogenes* in pork. This dataset was originally published by (Mataragas, Drosinos, Vaidanis, and Metaxopoulos 2006b). The data was saved as an **Excel** file using the resources provided by **ComBase** and are included as supplementary material to this article. The contents of the file can be loaded using `readxl::read_excel()`.

```
R> listeria_counts <- read_excel("listeria_combase.xlsx",
+    sheet = "Logcs") %>%
+    mutate(Time = as.numeric(Time), Logc = as.numeric(Logc))
R> temperatures <- read_excel("listeria_combase.xlsx",
+    sheet = "Temperatures") %>%
+    mutate(Time = as.numeric(Time), Temperature = as.numeric(Temperature))
R> pH_values <- read_excel("listeria_combase.xlsx", sheet = "pHs") %>%
+    mutate(Time = as.numeric(Time), pH = as.numeric(pH))
R> p1 <- ggplot(listeria_counts, aes(x = Time, y = Logc)) +
+    geom_point() +
+    theme(legend.position = "top")
R> p2 <- ggplot(temperatures, aes(x = Time, y = Temperature)) +
+    geom_line() +
+    theme(legend.position = "top")
R> p3 <- ggplot(pH_values, aes(x = Time, y = pH)) +
+    geom_line() +
+    theme(legend.position = "top")
R> plot_grid(p1, p2, p3)
```

When building a model based on data gathered under dynamic conditions, it is important to consider different hypotheses that may describe the observations. The microbial counts show the type of sigmoidal shape that can be described using primary models, with growth slowing down at the beginning and at the end of the experiment. If temperature was constant, this behavior could be attributed to the adaptation phase required before growth onset and the depletion of nutrients at the end of the exponential phase. However, the end of the "apparent" lag phase matches the moment when the temperature was raised from 4 to 7°C . In a similar way, the beginning of the "apparent" stationary phase takes place when temperature is lowered from 11 to 4°C. Considering that the minimum temperature for growth of *Listeria* spp. is slightly below 4°C (Mataragas, Drosinos, Siana, Skandamis, and Metaxopoulus 2006a), it is reasonable to assume that the apparent lag and stationary phases are not due to the mechanisms usually observed in isothermal experiments. Instead, they could be attributed to the effect of temperature on microbial growth. Furthermore, there is a slight increase in the population size during the first period at 4°C that is not observed in the second period at this temperature. This could be attributed to the effect of the pH on the growth rate, as this factor decreases from approximately 6.25 to 5.25 through the experiment (a common observation during the growth of *L. monocytogenes* due to the production of lactate). This effect could also be described using the gamma concept, defining a second gamma factor for pH.

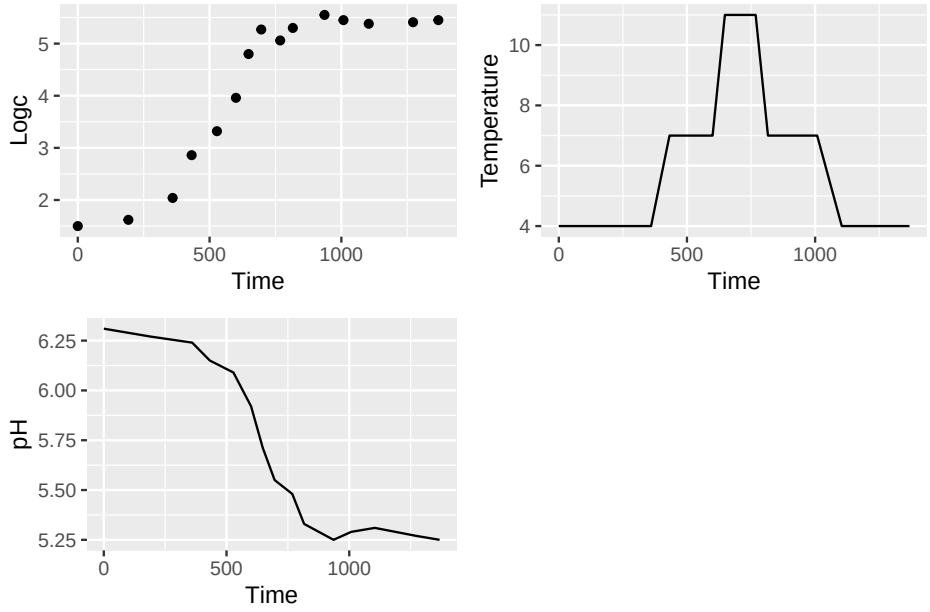

Figure 15: Illustration of the data retrieved from **ComBase** on the growth of *L. monocytogenes* in pork.

The `fit_growth()` function can be used to test whether these different hypotheses could explain the data. We can exploit the possibility to fix some parameters to fit a model that attributes the reduction in the growth rate to temperature changes. Then, we can use the S3 methods included in the package to evaluate whether the proposed models can describe the data. **ComBase** defines the temperature and pH profiles in two different **Excel** sheets, which must be joined before they can be used by **biogrowth**.

```
R> env_profile <- full_join(temperatures, pH_values,
+    by = c("Record ID", "Time")
+  )
```

Next, we need to define the secondary models. Due to its simplicity, we will use a Zwietering-type secondary model for both environmental factors.

```
R> sec_models <- list(Temperature = "Zwietering", pH = "Zwietering")
```

We can test different model hypotheses by fixing some model parameters to selected values. To test whether the initial lag phase is due to the temperature inhibition, we can fix the initial value of $Q(t)$ to a high value (`Q0 = 1e10`), resulting in a model without lag phase. In a similar way, we can fix the value of $N_{\max}$ to a value several orders of magnitude higher than the maximum microbial load one observed in the experiment (`Nmax = 1e8`), resulting in a model without a stationary phase.

Apart from that, we will fix the initial population size to the initial value of the observations. In predictive microbiology, the order of the model (n) is most often fixed to an integer value. In this model, the order of both secondary models is set to $n = 2$.

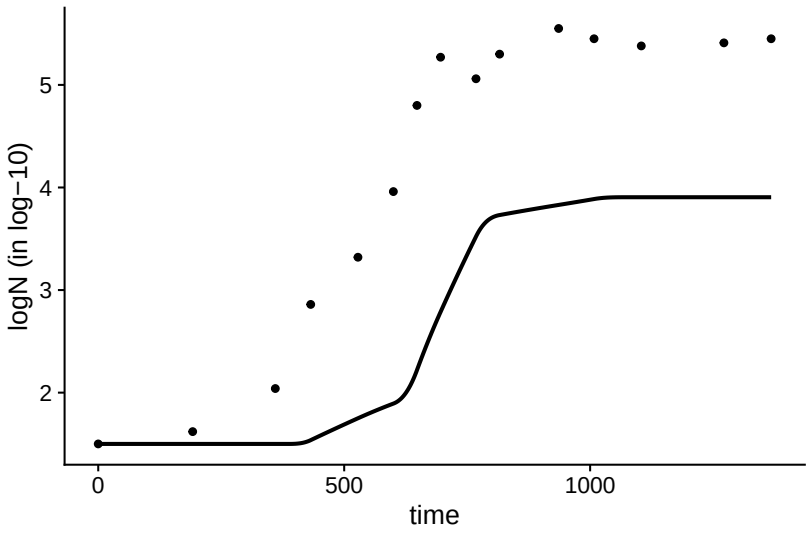

Figure 16: Initial guess for fitting the growth model to the growth of *L. monocytogenes* in pork observed under dynamic conditions.

```
R> logN0 <- min(listeria_counts$Logc)
R> known <- c(Q0 = 1e10, Temperature_n = 2, pH_n = 2, Nmax = 1e8,
+    N0 = 10^logN0)
```

The remaining model parameters (μ_{opt} and the cardinal parameters) are estimated from the dynamic data. Due to the use of regression, `fit_growth()` needs initial guesses for these parameters. In this example, we will use typical values for the growth of *L. monocytogenes*. Nonetheless, we can use `check_growth_guess()` to check that our initial guess is reasonably close to the data points. Note that, because **ComBase** does not use the default column names for **biogrowth**, they are defined using the `formula` argument.

```
R> start <- c(mu_opt = 1, Temperature_xopt = 35, Temperature_xmin = 5,
+    pH_xmin = 4, pH_xopt = 7)
R> check_growth_guess(listeria_counts, sec_models, c(start, known),
+    env_profile, environment = "dynamic", formula = Logc ~ Time)
```

The plot shows that the growth curve is reasonably close to the data points. Then, we can call `fit_growth()` with these arguments.

```
R> listeria_model <- fit_growth(environment = "dynamic", listeria_counts,
+    sec_models, start, known, env_conditions = env_profile,
+    formula = Logc ~ Time)
```

The instance of '`GrowthFit`' returned by `fit_growth()` implements several S3 methods to facilitate the analysis of the model fit. The `plot()` method can be used to validate that the model is able to describe the trend of the observations. In this case, the plot shows that the fitted model describe the trend of the experimental observations. Based on this, we can conclude that the data can be described considering only the effect of temperature and pH on

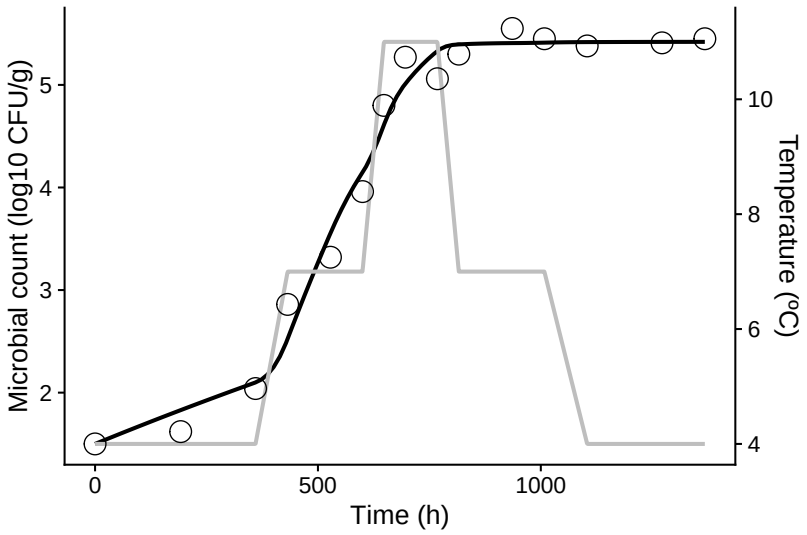

Figure 17: Fit of the Baranyi model with secondary model based on the gamma concept to the data on growth of *Listeria monocytogenes* under dynamic conditions (black). The temperature profile is shown as a gray line.

the growth rate, without introducing hypotheses related to the adaptation of cell metabolism (lag phase) or nutrient depletion (stationary phase).

```
R> plot(listeria_model, add_factor = "Temperature",
+    line_type2 = 1, line_col2 = "grey",
+    label_y1 = "Microbial count (log10 CFU/g)",
+    label_y2 = "Temperature (ºC)",
+    label_x = "Time (h)",
+    point_shape = 1, point_size = 5)
```

The summary S3 method can be used to retrieve the values of the parameter estimates and their standard errors.

```
R> summary(listeria_model)
```

```
Parameters:
                 Estimate Std. Error t value Pr(>|t|)
mu_opt             1.1669     0.5461   2.137 0.058330 .
Temperature_xopt  33.0219    27.8768   1.185 0.263578
Temperature_xmin   2.4712     0.4009   6.165 0.000106 ***
pH_xmin            5.2016     0.1039  50.068 2.44e-13 ***
pH_xopt            6.6221     1.0146   6.527 6.66e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2142 on 10 degrees of freedom
```

```
Parameter correlation:
                  mu_opt Temperature_xopt Temperature_xmin  pH_xmin  pH_xopt
mu_opt            1.0000           0.8461         -0.51629 -0.61135 -0.62524
Temperature_xopt  0.8461           1.0000         -0.27314 -0.28023 -0.93517
Temperature_xmin -0.5163          -0.2731          1.00000  0.76472 -0.03807
pH_xmin          -0.6113          -0.2802          0.76472  1.00000 -0.04806
pH_xopt          -0.6252          -0.9352         -0.03807 -0.04806  1.00000
```

It shows that there is extremely high uncertainty associated to the value of T_{opt} (relative standard error of 0.84). This could be attributed to the experimental design, which covers temperatures in a relatively short range (4 to 11°C), far from the parameter estimate (33°C). Another parameter with high uncertainty in this model is μ_{opt} (relative standard error of 0.47). This could be attributed to the high correlation between this parameter and T_{opt}, which inflates the uncertainty of both parameters. In order to reduce this uncertainty, we could fix parameter T_{opt} to 35°C, a value that has been reported for *Listeria* spp. in meat products (Liu, Wang, Liu, and Dong 2019). For the same reasons, we also fix pH_{opt} to 7.

```
R> known <- c(known, Temperature_xopt = 35, pH_xopt = 7)
R> start <- start[c("mu_opt", "Temperature_xmin", "pH_xmin")]
R> reduced_model <- fit_growth(environment = "dynamic",
+    listeria_counts, as.list(sec_models), start, known,
+    env_conditions = env_profile, formula = Logc ~ Time)
R> summary(reduced_model)

Parameters:
                 Estimate Std. Error  t value Pr(>|t|)
mu_opt           1.677883   0.214475    7.823 4.72e-06 ***
Temperature_xmin 2.189692   0.254278    8.611 1.76e-06 ***
pH_xmin          5.143662   0.001194 4308.702  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2004 on 12 degrees of freedom

Parameter correlation:
                  mu_opt Temperature_xmin  pH_xmin
mu_opt           1.00000          0.97696  0.01896
Temperature_xmin 0.97696          1.00000 -0.09215
pH_xmin          0.01896         -0.09215  1.00000
```

The summary table shows that fixing the model parameters results in a strong reduction of parameter uncertainty for μ_{opt} compared to the initial model. The `compare_growth_fits()` function can be used to assess whether this parameter reduction reduces the precision of the model fit. This function accepts a (named) list of models.

```
R> comparison_listeria <- compare_growth_fits(
+    list(Standard = listeria_model, Reduced = reduced_model)
+  )
```

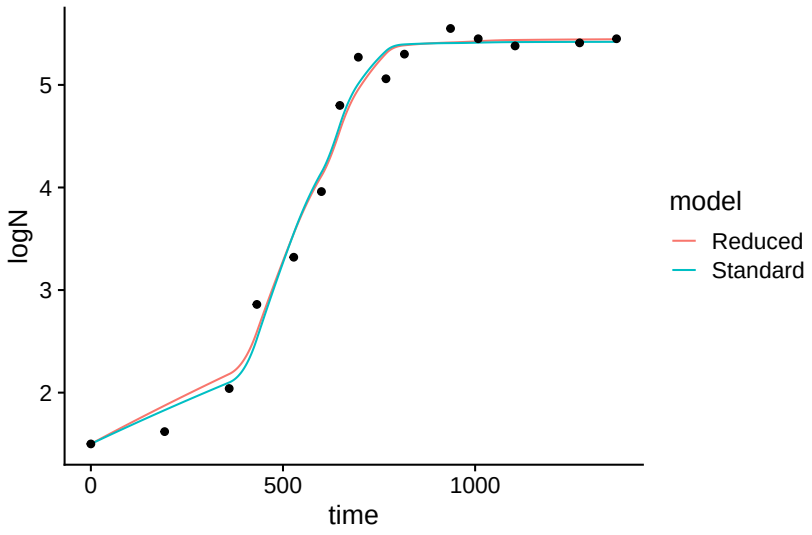

Figure 18: Comparison of the two model fits to the data on the growth of *L. monocytogenes* under dynamic conditions.

The `plot()` method compares both models fitted. In this case, it shows that there is practically no difference between the fitted curves.

```
R> plot(comparison_listeria)
```

Calling the `print()` method returns a table of the AIC for each model fit, showing that the "reduced" model is preferred over the standard one.

```
R> print(comparison_listeria)

Comparison between models fitted to data under isothermal conditions

Statistical indexes arranged by AIC:

# A tibble: 2 x 5
  model       AIC    df      ME  RMSE
  <chr>     <dbl> <int>   <dbl> <dbl>
1 Reduced  -0.469    12 0.0140  0.179
2 Standard  8.01     10 0.00706 0.175
```

An additional comparison that can be of interest, is the one provided by the `plot()` method when `type = 2`. This plot compares the expected values and standard errors of the parameter estimates for both models. The figure shows that both models practically estimate the same parameter values. However, the standard model estimates every parameter with higher uncertainty. This is often an indication of poor identifiability, being additional evidence that not all the parameters of the standard model can be identified with this experimental design.

```
R> plot(comparison_listeria, type = 2)
```

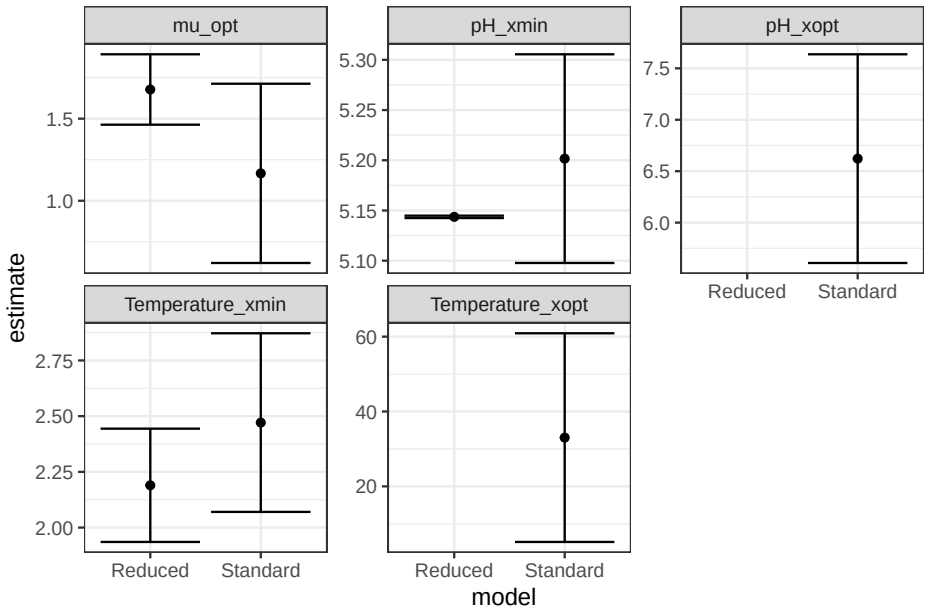

Figure 19: Comparison of the parameters estimates for the two models fitted to the data on the growth of *L. monocytogenes* under dynamic conditions. Note that the parameters that were fixed in the reduced model are not shown in their respective facets.

It is worth mentioning that the fitted model estimates a value of 5.14 for pH_{min}. This value is higher than the minimum pH enabling growth of *L. monocytogenes* at optimal temperature conditions, which usually ranges 4.4 (George, Lund, and Brocklehurst 1988). This type of deviation is typical when fitting a model to data gathered under dynamic conditions because experimental conditions rarely include sufficient data points in the vicinity of X_{min} or X_{max}. Therefore, the behavior of the population at extreme conditions has little leverage on the model fitted for many experimental designs. For this reason, in predictive microbiology, the parameters estimated in this approach are usually called "theoretical" X_{min} or X_{max} and must be further validated using dedicated growth boundary experiments (George *et al.* 1988; Le Marc, Huchet, Bourgeois, Guyonnet, Mafart, and Thuault 2002).

4.5. Growth models fitted to experiments under different conditions

In this last illustration, we show how a growth model defined based on primary and secondary models can be fitted to several experiments obtained under different environmental conditions using `fit_growth()`. For this illustration, we will also use data extracted from **ComBase** (**ComBase** ID: Lm_MpDyn_T1_R2). This dataset was originally published in the same article as the one used in the previous case study, and is included as supplementary material to this article.

```
R> listeria_counts2 <- read_excel("listeria_combase2.xlsx",
+    sheet = "Logcs") %>%
+    mutate(Time = as.numeric(Time), Logc = as.numeric(Logc))
R> temperatures2 <- read_excel("listeria_combase2.xlsx",
+    sheet = "Temperatures") %>%
```

```
+    mutate(Time = as.numeric(Time), Temperature = as.numeric(Temperature))
R> pH_values2 <- read_excel("listeria_combase2.xlsx", sheet = "pHs") %>%
+    mutate(Time = as.numeric(Time), pH = as.numeric(pH))
```

For global fitting, the `fit_growth()` function requires that the data is saved as a list, where each entry describes the information from each experiment. The microbial counts can be converted directly.

```
R> multiple_listeria <- list(R1 = listeria_counts, R2 = listeria_counts2)
```

Regarding the environmental conditions, in the same way as in the previous example, we need to put both temperature and pH in the same table before converting it into a list.

```
R> env_profile2 <- full_join(temperatures2, pH_values2,
+    by = c("Record ID", "Time")
+  )
R> multiple_profile <- list(R1 = env_profile, R2 = env_profile2)
```

Now, we can fit the growth model using `fit_growth()`. In this case, we pass the argument `approach = "global"` to indicate that we are fitting a unique model to different growth experiments. We also use a Monte Carlo algorithm because we later intend to make predictions including parameter uncertainty. Note that, when calling the function we are passing additional arguments to `FME::modMCMC()`. Namely, the number of iterations for updating the covariance matrix and the burn-in length.

```
R> set.seed(12421)
R> global_fit <- fit_growth(multiple_listeria, sec_models, start,
+    known, environment = "dynamic", algorithm = "MCMC",
+    approach = "global", env_conditions = multiple_profile,
+    formula = Logc ~ Time, niter = 6000, updatecov = 500,
+    burninlength = 1000
+  )

number of accepted runs: 648 out of 6000 (10.8%)
```

Although the algorithm converges to a solution, the percentage of accepted runs is relatively low. This often indicates that the convergence of the algorithm could be improved by modifying its parameters. Nonetheless, this topic is highly specific and is out of the scope of this article. The interested reader is referred to the specific literature on the topic (e.g. (Brooks, Gelman, Jones, and Meng 2011)).

We can call the `plot()` method of the instance of `GlobalGrowthFit` returned by the function to check that the fitted model properly describes the trend in both experiments.

```
R> plot(global_fit, add_factor = "pH", label_y2 = "pH value",
+    label_y1 = "Microbial concentration (log CFU/g)",
+    label_x = "Time (h)", line_col2 = "maroon", line_type2 = 1)
```

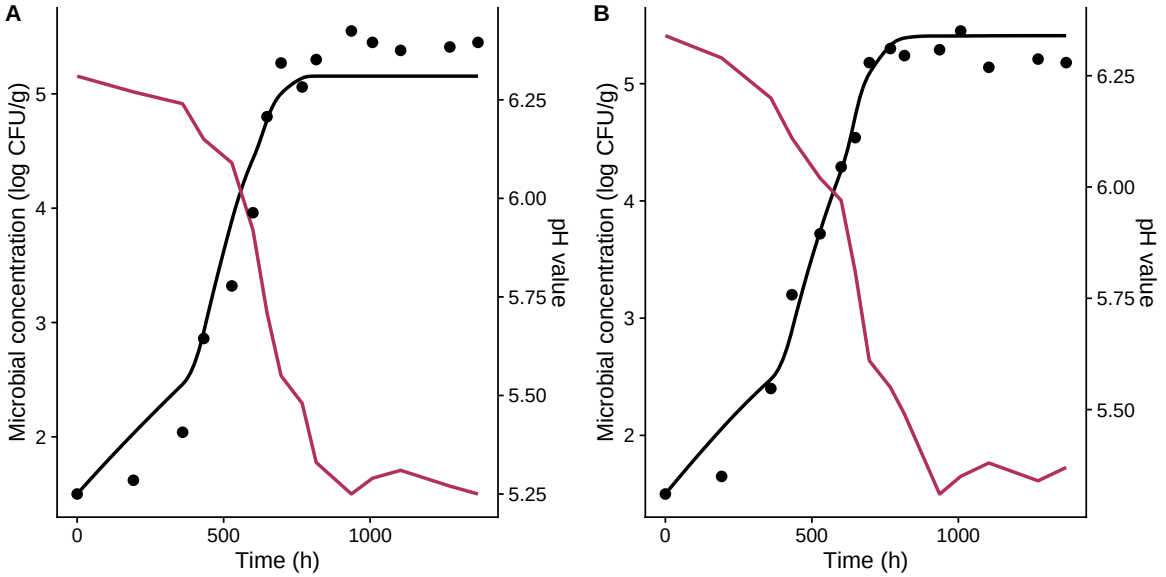

Figure 20: Global fit of the growth model for *L. monocytogenes* in pork meat from two independent experiments (black line). The pH profile is shown in maroon.

This figure illustrates that the fitted model can describe the variation in the population size observed in both experiments. This is further support of the hypothesis that the sigmoidal shape in the population size is due to the effect of the environmental conditions rather than the usual causes for a lag or stationary phases.

Then, we can use the fitted model for making predictions. In this illustration, we will include parameter uncertainty in the predictions using `predictMCMC()`. This prediction can be made for any type of environmental condition, constant or not. As an example, we will use an environmental profile with constant temperature (5°C) and pH decaying from pH 7 to pH 5 linearly over 1,000 hours. This functions also allows modifying any model parameter through the `newpars` argument. Hence, we will also modify the initial population size (N_0) to 1 CFU/g, to represent conditions different to those used under laboratory conditions.

```
R> pred_cond <- tibble(time = c(0, 1000), Temperature = c(5),
+    pH = c(7, 5))
R> set.seed(14241)
R> pred_uncertainty <- predictMCMC(global_fit, seq(0, 1000, length = 100),
+    pred_cond, niter = 1000, formula = . ~ time, newpars = list(N0 = 1)
+  )
```

The instance of '`MCMCgrowth`' includes several S3 methods to facilitate the interpretation of the results. The plot method can be used to visualize the effect of parameter uncertainty in the model predictions.

```
R> plot(pred_uncertainty, add_factor = "pH",
+    label_y1 = "Microbial concentration (log CFU/g)",
+    label_y2 = "pH value", label_x = "Time (h)",
+    fill_80 = "lightblue", line_col = "darkblue")
```

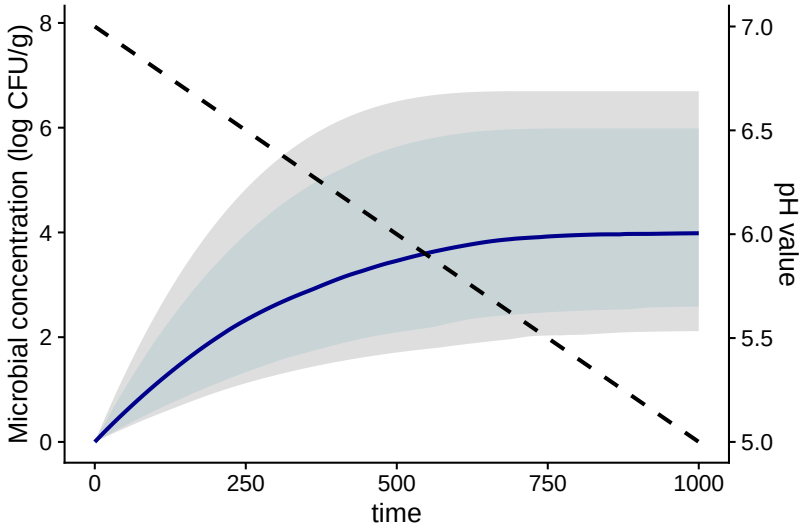

Figure 21: Prediction band for the growth of *L. monocytogenes* under dynamic environmental conditions including parameter uncertainty. The grey and blue ribbons represent, respectively, the credible intervals at the 0.9 and 0.8 levels. The dashed, black line represents the pH profile.

Once this prediction has been calculated, we can also estimate the distribution of the time required to reach a population size of 2 log CFU/g. This value has been selected because it is often used as a microbiological criterion to define the shelf life of food products.

```
R> time_to_100 <- time_to_size(pred_uncertainty, 2, type = "distribution")
R> time_to_100

Distribution of the time required to reach a target population size

# A tibble: 1 x 5
  m_time sd_time med_time   q10   q90
   <dbl>   <dbl>    <dbl> <dbl> <dbl>
1   231.    141.     192.  101.  414.
```

Note that passing `type = "distribution"` calculates a distribution for the elapsed time instead of a discrete value. This distribution represents the uncertainty in the parameter estimates, and can also be represented as a histogram using the S3 method for `plot()`.

```
R> plot(time_to_100)

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 38 rows containing non-finite values (stat_bin).
```

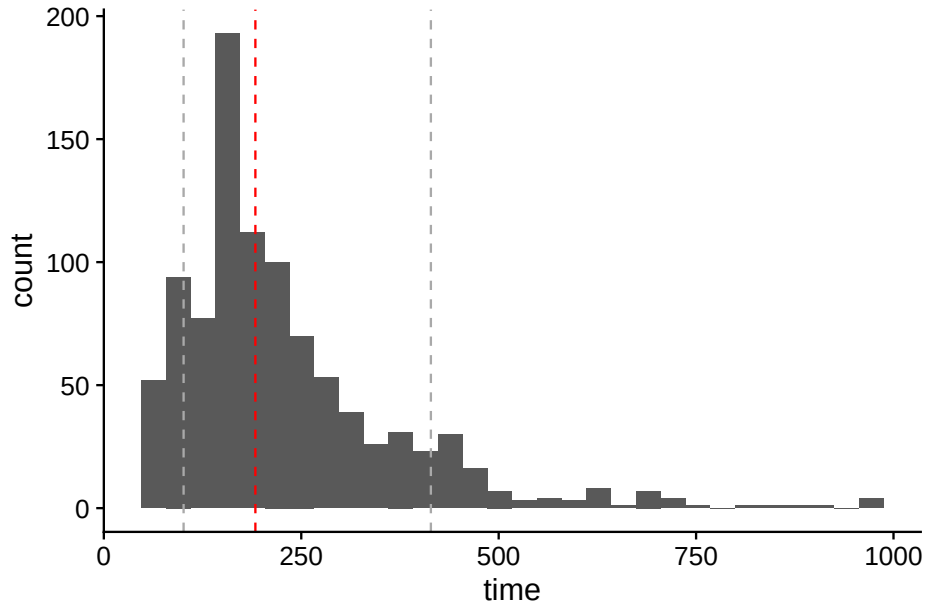

Figure 22: Estimated distribution for the time to reach 100 CFU/g for *L. monocytogenes*. The vertical, red line represents the median of the simulations, whereas the vertical, grey lines show the 10th and 90th percentiles.

5. Closing remarks

Mathematical models are applied in a large variety of fields to describe the growth of populations. However, their application usually requires the use of advanced mathematical methods to build the model (e.g. inference) and/or to make prediction (e.g. solving differential equations). This can be specially troublesome when analyzing growth under dynamic environmental conditions. The **biogrowth** package can make these models more accessible, providing an R interface to the functions required for building the models and making predictions. Although the methods implemented in the package are based on predictive microbiology, the mathematical equations and methods are also applied to a large variety of fields, such as biotechnology or economy.

Nonetheless, the **biogrowth** package also has some limitations. This is due to the fact that the effect of changes in the environmental conditions on the duration of the lag phase is highly complex, depending both on factors of the population (e.g. physiological state of the cells) and external factors (e.g., temperature or pH). Although some models have been published to describe this effect (Augustin, Rosso, and Carlier 2000; Yue *et al.* 2019; Delignette-Muller, Baty, Cornu, and Bergis 2005), they are quite specific and hard to extrapolate to non-laboratory conditions. Consequently, these modeling approaches are not included in **biogrowth**. For the same reason, the package does not include the possibility to define interactions between gamma factors. Another limitation of the package is that it does not include the possibility to implement user-defined models. The reason for this is that it is designed as a tool to facilitate the application of growth models that are broadly applied in the literature. We believe that more general packages for model fitting (e.g. **FME**) are more suited for tuning model equations. Nonetheless, for advanced users, the wiki from its GitHub page

(https://github.com/albgarre/biogrowth) explains how to extend the package with new primary and secondary models.

The fact that **biogrowth** is based on parametric models also brings limitations with respect to already existing tools. These models are more "rigid" than non-parametric ones, being able to describe only sigmoidal curves. Therefore, if the experimental data deviates from this shape (e.g., when analyzing data from a bioscreen), packages based on non-parametric models (e.g., smoothing splines) would often be more suitable (e.g. **growthrates** or **growthcurver**). In a similar way, we advised against the use of **biogrowth** to describe inactivation (i.e., decreasing curves) because the models included in the package were based on hypotheses related to the growth of population. Instead, non-parametric approaches or packages including inactivation models (e.g., **bioinactivation**, Garre *et al.* 2017) are advised. On the other hand, the parametric models provide a more mechanistic explanation of the population growth; e.g. providing estimates of the lag phase duration or the relationship between the environmental factors and μ. Therefore, it can be of greater interest in cases aiming at understanding the state of the system and predicting its future state rather than just fitting the experimental data. Furthermore, as already mentioned, the growth models implemented in the package are prone to parameter identifiability issues. The functions included in the package can be a basis for future studies on model identifiability and optimal experiment design based on local sensitivities and the optimization of the Fisher Information Matrix (Garre, González-Tejedor, Peñalver-Soto, Fernández, and Egea 2018b).

As a closing comment, one point to consider when using the functions included in this package is that the more complex tools are not better than simple ones in every case study (Zwietering 2009). In most situations, models defined for dynamic environmental conditions will make additional hypotheses than an equivalent model defined for constant conditions. On top of that, these hypotheses are often harder to validate experimentally. For instance, dynamic growth models make the hypothesis that cells respond instantly to changes in the environmental conditions, a hypothesis that may be false in some scenarios (Antolinos, Muñoz-Cuevas, Ros-Chumillas, Periago, Fernández, and Le Marc 2012; Garre, Egea, Iguaz, Palop, and Fernandez 2018a). Therefore, it is important that the analyst considers and compares all the functions offered by the software, selecting the one that is more appropriate for the case studied.

Acknowledgements

The authors would like to thank Prof. Pablo S. Fernandez from Technical University of Cartagena; Prof. Fernando Perez-Rodriguez, Dr. Aricia Possas and Ms. Cristina Diaz Martinez from University of Cordoba; Dr. Ignacio Alvarez Lanzarote and Dr. Guillermo Cebrian from University of Zaragoza; and Dr. Annemarie Pielaat and Dr. Joost Smid from Unilever for reviewing and providing valuable feedback on early versions of the software. They also thank Dr. Sara Bover-Cid, from IRTA for sharing the data regarding the temperature profiles in refrigerators.

Alberto Garre was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Individual Fellowship grant No 844423 (FANTASTICAL).

References

Adam JA, Bellomo N (2012). *A Survey of Models for Tumor-Immune System Dynamics*. Springer-Verlag. doi:10.1007/978-0-8176-8119-7.

Adler A (2023). ***lamW**: Lambert-W Function*. R package version 2.1.2, URL https://CRAN.R-project.org/package=lamW.

Antolinos V, Muñoz-Cuevas M, Ros-Chumillas M, Periago PM, Fernández PS, Le Marc Y (2012). "Modelling the Effects of Temperature and Osmotic Shifts on the Growth Kinetics of *Bacillus weihenstephanensis* in Broth and Food Products." *International Journal of Food Microbiology*, **158**(1), 36–41. doi:10.1016/j.ijfoodmicro.2012.06.017.

Augustin JC, Rosso L, Carlier V (2000). "A Model Describing the Effect of Temperature History on Lag Time for *Listeria Monocytogenes*." *International Journal of Food Microbiology*, **57**(3), 169–181. doi:10.1016/s0168-1605(00)00260-9.

Baranyi J, Roberts TA (1994). "A Dynamic Approach to Predicting Bacterial Growth in Food." *International Journal of Food Microbiology*, **23**(3-4), 277–294. doi:10.1016/0168-1605(94)90157-0.

Baranyi J, Tamplin ML (2004). "ComBase: A Common Database on Microbial Responses to Food Environments." *Journal of Food Protection*, **67**(9), 1967–1971. doi:10.4315/0362-028x-67.9.1967.

Baty F, Delignette-Muller ML, Siberchicot A (2021). ***nlsMicrobio**: Data Sets and Nonlinear Regression Models Dedicated to Predictive Microbiology*. R package version 0.0-3, URL https://CRAN.R-project.org/package=nlsMicrobio.

Baty F, Ritz C, Charles S, Brutsche M, Flandrois JP, Delignette-Muller ML (2015). "A Toolbox for Nonlinear Regression in R: The Package **nlstools**." *Journal of Statistical Software*, **66**(5), 1–21. doi:10.18637/jss.v066.i05.

Brooks S, Gelman A, Jones G, Meng XL (2011). *Handbook of Markov Chain Monte Carlo*. CRC press. doi:10.1201/b10905.

Buchanan RL, Whiting RC (1996). "Risk Assessment and Predictive Microbiology." *Journal of Food Protection*, **59**(13), 31–36. doi:10.4315/0362-028x-59.13.31.

Buchanan RL, Whiting RC, Damert WC (1997). "When Is Simple Good Enough: A Comparison of the Gompertz, Baranyi, and Three-Phase Linear Models for Fitting Bacterial Growth Curves." *Food Microbiology*, **14**(4), 313–326. doi:10.1006/fmic.1997.0125.

Cabecinhas M, Domingues P, Sampaio P, Bernardo M, Franceschini F, Galetto M, Gianni M, Gotzamani K, Mastrogiacomo L, Hernandez-Vivanco A (2018). "Integrated Management Systems Diffusion Models in South European Countries." *International Journal of Quality & Reliability Management*, **35**(10), 2289–2303. doi:10.1108/ijqrm-03-2017-0044.

Carlin F, Albagnac C, Rida A, Guinebretière MH, Couvert O, Nguyen-the C (2013). "Variation of Cardinal Growth Parameters and Growth Limits According to Phylogenetic Affiliation in the *Bacillus cereus* Group. Consequences for Risk Assessment." *Food Microbiology*, **33**(1), 69–76. doi:10.1016/j.fm.2012.08.014.

Delignette-Muller ML, Baty F, Cornu M, Bergis H (2005). "Modelling the Effect of a Temperature Shift on the Lag Phase Duration of *Listeria monocytogenes*." *International Journal of Food Microbiology*, **100**(1), 77–84. doi:10.1016/j.ijfoodmicro.2004.10.021.

EFSA Scientific Committee, Benford D, Halldorsson T, Jeger MJ, Knutsen HK, More S, Naegeli H, Noteborn H, Ockleford C, Ricci A, Rychen G, Schlatter JR, Silano V, Solecki R, Turck D, Younes M, Craig P, Hart A, Von Goetz N, Koutsoumanis K, Mortensen A, Ossendorp B, Martino L, Merten C, Mosbach-Schulz O, Hardy A (2018). "Guidance on Uncertainty Analysis in Scientific Assessments." *EFSA Journal*, **16**(1). doi:10.2903/j.efsa.2018.5123.

Fernandez-Ricaud L, Kourtchenko O, Zackrisson M, Warringer J, Blomberg A (2016). "**PRECOG**: A Tool for Automated Extraction and Visualization of Fitness Components in Microbial Growth Phenomics." *BMC Bioinformatics*, **17**(1), 249. doi:10.1186/s12859-016-1134-2.

García MR, Vilas C, Herrera JR, Bernárdez M, Balsa-Canto E, Alonso AA (2015). "Quality and Shelf-Life Prediction for Retail Fresh Hake (*Merluccius merluccius*)." *International Journal of Food Microbiology*, **208**, 65–74. doi:10.1016/j.ijfoodmicro.2015.05.012.

Garre A, Egea JA, Iguaz A, Palop A, Fernandez PS (2018a). "Relevance of the Induced Stress Resistance When Identifying the Critical Microorganism for Microbial Risk Assessment." *Frontiers in Microbiology*, **9**, 1663. doi:10.3389/fmicb.2018.01663.

Garre A, Fernández PS, Lindqvist R, Egea JA (2017). "**bioinactivation**: Software for Modelling Dynamic Microbial Inactivation." *Food Research International*, **93**, 66–74. doi:10.1016/j.foodres.2017.01.012.

Garre A, González-Tejedor G, Peñalver-Soto JL, Fernández PS, Egea JA (2018b). "Optimal Characterization of Thermal Microbial Inactivation Simulating Non-Isothermal Processes." *Food Research International*, **107**, 267–274. doi:10.1016/j.foodres.2018.02.040.

Garre A, Koomen J, Den Besten HMW, Zwietering MH (2023). ***biogrowth**: Modelling of Population Growth*. R package version 1.0.3, URL https://CRAN.R-project.org/package=biogrowth.

George SM, Lund BM, Brocklehurst TF (1988). "The Effect of pH and Temperature on Initiation of Growth of *Listeria Monocytogenes*." *Letters in Applied Microbiology*, **6**(6), 153–156. doi:10.1111/j.1472-765x.1988.tb01237.x.

Gonzales RR, Kim JS, Kim SH (2019). "Optimization of Dilute Acid and Enzymatic Hydrolysis for Dark Fermentative Hydrogen Production from the Empty Fruit Bunch of Oil Palm." *International Journal of Hydrogen Energy*, **44**(4), 2191–2202. doi:10.1016/j.ijhydene.2018.08.022.

González-Tejedor GA, Garre A, Esnoz A, Artés-Hernández F, Fernández PS (2018). "Effect of Storage Conditions in the Response of *Listeria monocytogenes* in a Fresh Purple Vegetable Smoothie Compared with an Acidified TSB Medium." *Food Microbiology*, **72**, 98–105. doi:10.1016/j.fm.2017.11.005.

Gupta BM, Kumar S, Sangam SL, Karisiddappa CR (2002). "Modeling the Growth of World Social Science Literature." *Scientometrics*, **53**(1), 161–164. doi:10.1023/a:1014844222898.

Haario H, Laine M, Mira A, Saksman E (2006). "DRAM: Efficient Adaptive MCMC." *Statistics and Computing*, **16**(4), 339–354. doi:10.1007/s11222-006-9438-0.

Henry L, Wickham H (2022). ***lifecycle**: Manage the Life Cycle of Your Package Functions*. R package version 1.0.3, URL https://CRAN.R-project.org/package=lifecycle.

Huang L, Li C (2020). "Growth of *Clostridium Perfringens* in Cooked Chicken During Cooling: One-Step Dynamic Inverse Analysis, Sensitivity Analysis, and Markov Chain Monte Carlo Simulation." *Food Microbiology*, **85**, 103285. doi:10.1016/j.fm.2019.103285.

Jiang L, He X, Jin Y, Ye M, Sang M, Chen N, Zhu J, Zhang Z, Li J, Wu R (2018). "A Mapping Framework of Competition – Cooperation QTLs That Drive Community Dynamics." *Nature Communications*, **9**(1), 3010. doi:10.1038/s41467-018-05416-w.

Jofré A, Latorre-Moratalla ML, Garriga M, Bover-Cid S (2019). "Domestic Refrigerator Temperatures in Spain: Assessment of Its Impact on the Safety and Shelf-Life of Cooked Meat Products." *Food Research International*, **126**, 108578. doi:10.1016/j.foodres.2019.108578.

Kahm M, Hasenbrink G, Lichtenberg-Fraté H, Ludwig J, Kschischo M (2010). "**grofit**: Fitting Biological Growth Curves with R." *Journal of Statistical Software*, **33**(1), 1–21. doi:10.18637/jss.v033.i07.

Kamrad S, Rodríguez-López M, Cotobal C, Correia-Melo C, Ralser M, Bähler J (2020). "**Pyphe**, A Python Toolbox for Assessing Microbial Growth and Cell Viability in High-Throughput Colony Screens." *eLife*, **9**, e55160. doi:10.7554/elife.55160.

Le Marc Y, Huchet V, Bourgeois CM, Guyonnet JP, Mafart P, Thuault D (2002). "Modelling the Growth Kinetics of *Listeria* as a Function of Temperature, pH and Organic Acid Concentration." *International Journal of Food Microbiology*, **73**(2), 219–237. doi:10.1016/s0168-1605(01)00640-7.

Liu Y, Wang X, Liu B, Dong Q (2019). "One-Step Analysis for *Listeria Monocytogenes* Growth in Ready-to-Eat Braised Beef at Dynamic and Static Conditions." *Journal of Food Protection*, **82**(11), 1820–1827. doi:10.4315/0362-028x.jfp-18-574.

Martínez-Hernández GB, Taboada-Rodríguez A, Garre A, Marín-Iniesta F, López-Gómez A (2021). "The Application of Essential Oil Vapors at the End of Vacuum Cooling of Fresh Culinary Herbs Promotes Aromatic Recovery." *Foods*, **10**(3), 498. doi:10.3390/foods10030498.

Mataragas M, Drosinos EH, Siana P, Skandamis P, Metaxopoulus I (2006a). "Determination of the Growth Limits and Kinetic Behavior of *Listeria monocytogenes* in a Sliced Cooked Cured Meat Product: Validation of the Predictive Growth Model under Constant and Dynamic Temperature Storage Conditions." *Journal of Food Protection*, **69**(6), 1312–1321. doi:10.4315/0362-028x-69.6.1312.

Mataragas M, Drosinos EH, Vaidanis A, Metaxopoulos I (2006b). "Development of a Predictive Model for Spoilage of Cooked Cured Meat Products and Its Validation under Constant and Dynamic Temperature Storage Conditions." *Journal of Food Science*, **71**(6), M157–M167. doi:10.1111/j.1750-3841.2006.00058.x.

Nguimkeu P (2014). "A Simple Selection Test between the Gompertz and Logistic Growth Models." *Technological Forecasting and Social Change*, **88**, 98–105. doi:10.1016/j.techfore.2014.06.017.

Notebaart RA, Kintses B, Feist AM, Papp B (2018). "Underground Metabolism: Network-Level Perspective and Biotechnological Potential." *Current Opinion in Biotechnology*, **49**, 108–114. doi:10.1016/j.copbio.2017.07.015.

Öksüz HB, Buzrul S (2020). "Monte Carlo Analysis for Microbial Growth Curves." *Journal of Microbiology, Biotechnology and Food Sciences*, **10**(3), 418–423. doi:10.15414/jmbfs.2020.10.3.418-423.

Perez DR (2023). ***growthmodels**: Nonlinear Growth Models*. R package version 1.3.1, URL https://CRAN.R-project.org/package=growthmodels.

Perez-Rodriguez F, Valero A (2012). *Predictive Microbiology in Foods*. Springer-Verlag.

Petzoldt T (2022). ***growthrates**: Estimate Growth Rates from Experimental Data*. R package version 0.8.4, URL https://CRAN.R-project.org/package=growthrates.

Ratkowsky DA, Lowry RK, McMeekin TA, Stokes AN, Chandler RE (1983). "Model for Bacterial Culture Growth Rate throughout the Entire Biokinetic Temperature Range." *Journal of Bacteriology*, **154**(3), 1222–1226. doi:10.1128/jb.154.3.1222-1226.1983.

Ratkowsky DA, Olley J, McMeekin TA, Ball A (1982). "Relationship between Temperature and Growth Rate of Bacterial Cultures." *Journal of Bacteriology*, **149**(1), 1–5. doi:10.1128/jb.149.1.1-5.1982.

Ritz C, Baty F, Streibig JC, Gerhard D (2015). "Dose-Response Analysis Using R." *PLOS One*, **10**(12), e0146021. doi:10.1371/journal.pone.0146021.

Rosso L, Lobry JR, Bajard S, Flandrois JP (1995). "Convenient Model to Describe the Combined Effects of Temperature and pH on Microbial Growth." *Applied and Environmental Microbiology*, **61**(2), 610–616. doi:10.1128/aem.61.2.610-616.1995.

Schendel T, Jung C, Lindtner O, Greiner M (2018). "Guidelines for Uncertainty Analysis: Application of the respective Documents of EFSA and BfR for Exposure Assessments." *EFSA Supporting Publications*, **15**(7), 1472E. doi:10.2903/sp.efsa.2018.en-1472.

Schmidt PJ, Emelko MB, Thompson ME (2019). "Recognizing Structural Nonidentifiability: When Experiments Do Not Provide Information about Important Parameters and Misleading Models Can Still Have Great Fit." *Risk Analysis*, **40**(2), 352–369. doi:10.1111/risa.13386.

Soetaert K, Petzoldt T (2010). "Inverse Modelling, Sensitivity and Monte Carlo Analysis in R Using Package **FME**." *Journal of Statistical Software*, **33**(3), 1–28. doi:10.18637/jss.v033.i03.

Soetaert K, Petzoldt T, Setzer RW (2010). "Solving Differential Equations in R: Package **deSolve**." *Journal of Statistical Software*, **33**(9), 1–25. doi:10.18637/jss.v033.i09.

Spada G, Melini D (2020). "Evolution of the Number of Communicative Civilizations in The Galaxy: Implications on Fermi Paradox." *International Journal of Astrobiology*, **19**(4), 314–319. doi:10.1017/s1473550420000063.

Sprouffske K, Wagner A (2016). "**growthcurver**: An R Package for Obtaining Interpretable Metrics from Microbial Growth Curves." *BMC Bioinformatics*, **17**(1), 172. doi:10.1186/s12859-016-1016-7.

Telen D, Logist F, Van Derlinden E, Tack I, Van Impe J (2012). "Optimal Experiment Design for Dynamic Bioprocesses: A Multi-Objective Approach." *Chemical Engineering Science*, **78**, 82–97. doi:10.1016/j.ces.2012.05.002.

Tsai BH (2013). "Predicting the Diffusion of LCD TVs by Incorporating Price in the Extended Gompertz Model." *Technological Forecasting and Social Change*, **80**(1), 106–131. doi:10.1016/j.techfore.2012.07.006.

Vásquez GA, Busschaert P, Haberbeck LU, Uyttendaele M, Geeraerd AH (2014). "An Educationally Inspired Illustration of Two-Dimensional Quantitative Microbiological Risk Assessment (QMRA) and Sensitivity Analysis." *International Journal of Food Microbiology*, **190**, 31–43. doi:10.1016/j.ijfoodmicro.2014.07.034.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York. URL https://www.stats.ox.ac.uk/pub/MASS4/.

Verhulst PF (1838). "Notice sur la loi que la population suit dans son accroissement." *Correspondance Mathématique et Physique*, **10**, 113–121.

Veríssimo A, Paixão L, Neves AR, Vinga S (2013). "**BGFit**: Management and Automated Fitting of Biological Growth Curves." *BMC Bioinformatics*, **14**(1), 283. doi:10.1186/1471-2105-14-283.

Vilas C, Arias-Mendez A, Garcia MR, Alonso AA, Balsa-Canto E (2018). "Toward Predictive Food Process Models: A Protocol for Parameter Estimation." *Critical Reviews in Food Science and Nutrition*, **58**(3), 436–449. doi:10.1080/10408398.2016.1186591.

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the **tidyverse**." *Journal of Open Source Software*, **4**(43), 1686. doi:10.21105/joss.01686.

Wickham H, Danenberg P, Csárdi G, Eugster M (2022a). ***roxygen2***: *In-Line Documentation for R*. R package version 7.2.3, URL https://CRAN.R-project.org/package=roxygen2.

Wickham H, Hester J, Chang W, Bryan J (2022b). ***devtools***: *Tools to Make Developing R Packages Easier*. R package version 2.4.5, URL https://CRAN.R-project.org/package=devtools.

Wilke CO (2020). ***cowplot**: Streamlined Plot Theme and Plot Annotations for **ggplot2***. R package version 1.1.1, URL https://CRAN.R-project.org/package=cowplot.

Xie Y (2015). *Dynamic Documents with R and **knitr***. Chapman & Hall/CRC, Boca Raton.

Yue S, Liu Y, Wang X, Xu D, Qiu J, Liu Q, Dong Q (2019). "Modeling the Effects of the Preculture Temperature on the Lag Phase of *Listeria Monocytogenes* at 25°C." *Journal of Food Protection*, **82**(12), 2100–2107. doi:10.4315/0362-028x.jfp-19-117.

Zwietering MH (2009). "Quantitative Risk Assessment: Is More Complex Always Better? Simple Is Not Stupid and Complex Is Not Always More Correct." *International Journal of Food Microbiology*, **134**(1-2), 57–62. doi:10.1016/j.ijfoodmicro.2008.12.025.

Zwietering MH, De Wit JC, Cuppers HGAM, Van 't Riet K (1994). "Modeling of Bacterial Growth with Shifts in Temperature." *Applied and Environmental Microbiology*, **60**(1), 204–213. doi:10.1128/aem.60.1.204-213.1994.

Zwietering MH, Jongenburger I, Rombouts FM, Van 't Riet K (1990). "Modeling of the Bacterial Growth Curve." *Applied and Environmental Microbiology*, **56**(6), 1875–1881. doi:10.1128/aem.56.6.1875-1881.1990.

Zwietering MH, Wijtzes T, De Wit JC, Van 't Riet K (1992). "A Decision Support System for Prediction of the Microbial Spoilage in Foods." *Journal of Food Protection*, **55**(12), 973–979. doi:10.4315/0362-028x-55.12.973.

Affiliation:

Alberto Garre
Food Microbiology
Wageningen University & Research
and
Department of Agronomical Engineering & Institute of Plant Biotechnology
Universidad Politécnica de Cartagena
Murcia, Paseo Alfonso XIII, 48, 30203, Spain
E-mail: alberto.garre@upct.es

Jeroen Koomen, Heidy M. W. den Besten, Marcel H. Zwietering
Food Microbiology
Wageningen University & Research
P.O. Box 17, 6700 AA Wageningen, The Netherlands
E-mail: jeroen.koomen@wur.nl, heidy.denbesten@wur.nl, marcel.zwietering@wur.nl

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
September 2023, Volume 107, Issue 1 *Submitted:* 2021-09-22
doi:10.18637/jss.v107.i01 *Accepted:* 2023-02-24