grofit: Fitting Biological Growth Curves with R

Matthias Kahm
RheinAhrCampus

Guido Hasenbrink
University of Bonn

Hella Lichtenberg-Fraté
University of Bonn

Jost Ludwig
University of Bonn

Maik Kschischo
RheinAhrCampus

Abstract

The following description of the package **grofit** was also published as Kahm *et al.* (2010). The **grofit** package was developed to fit many growth curves obtained under different conditions in order to derive a conclusive dose-response curve, for instance for a compound that potentially affects growth. **grofit** fits data to different parametric models and in addition provides a model free spline method to circumvent systematic errors that might occur within application of parametric methods. This amendment increases the reliability of the characteristic parameters (e.g.,lag phase, maximal growth rate, stationary phase) derived from a single growth curve. By relating obtained parameters to the respective condition (e.g.,concentration of a compound) a dose response curve can be derived that enables the calculation of descriptive pharma-/toxicological values like half maximum effective concentration (EC50). Bootstrap and cross-validation techniques are used for estimating confidence intervals of all derived parameters.

Keywords: growth curve, dose response curve, EC50, bootstrap.

1. Introduction

Modeling biological growth pertains to several hierarchically ordered complexity levels like cells, organisms, populations and communities or ecosystems. Approaches to understanding patterns of change comprise but are not limited to issues as e.g., cellular growth rates (Airoldi *et al.* 2009), coordination of growth with cell division (Alarcorn and Tindall 2007), growth and metamorphosis within development and life cycles at the organismal level (Qu *et al.* 2004) birth and survival rates or variations within species at the population level and not least living ecosystems and growth (Jorgensen *et al.* 2000; Fath *et al.* 2004) or food chains. Compared to kinetic analyses of biochemical data such issues are oriented towards the understanding of the mechanisms that link two or more processes, like e.g., the relationships between growth and cell cycle have been fitted using empirical functions or, more recently, by mathematical modeling based on a signal transduction network (Qu *et al.* 2004). In contrast, modeling of biochemical data pertain to a single or few biochemical components or enzyme kinetic studies, the narrowest/lowest functional level within the system properties of a cell. Obtained data sets require in depth parametric statistics to estimate the precision of the results involving also descriptive methods and virtually all of the models needed for kinetic studies are non-linear.

Biologists often utilize growth experiments to analyse basic properties of a given organism or

cellular model. To investigate the specific effect of a given experimental set up or condition, e.g., a compound or substrate, characteristic parameters of the growth curves are derived. This should ideally reveal a relationship between the concentration of a compound/substrate and its corresponding effect on a particular growth parameter. Having obtained a statistical relevant number of such growth curves (under different conditions like compound/substrate concentrations) dose response plots can be computed that enable the estimation of characteristic descriptive values such as EC50, IC50 (half maximum effective or inhibitory concentration) or else.

A typical example of the workflow implemented in **grofit** is given in Figure 1. Yeast cells were treated with different concentrations of a compound (here Hygromycin B). Growth was measured as the optical density (see (Hasenbrink *et al.* 2006) for details) at different time points. From the fitted growth curves the maximum growth rate μ (see Figure 2) was derived. The response μ was then plotted against the dose in Figure 1(b). A dose response curve was fitted and the EC50 value was estimated (see Section 4).

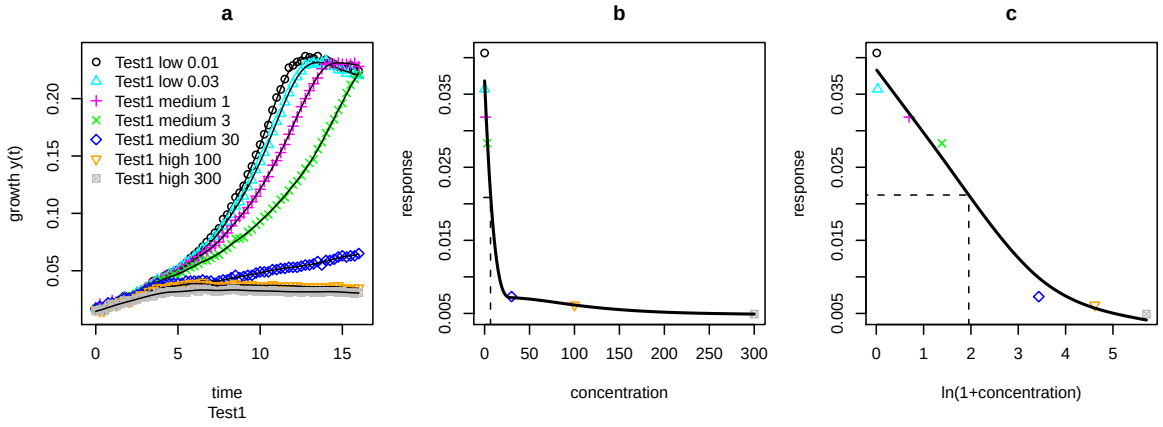

Figure 1: Deriving dose response curves from growth experiments: (a) Several fitted growth curves obtained under different concentrations (in μM) of Hygromycin B. (b) The maximum slope corresponding to the growth rate μ (see Figure 2) of each curve in (a) is calculated and plotted *vs.* the corresponding concentration. From these data points a dose response curve is estimated by fitting a smoothed spline. Consequently, the EC50 value 6.92 μM is estimated. (c) In order to obtain a more uniform distribution of the data points a logarithmic transformation to the concentration axis can be applied.

Many different mathematical models for growth have been developed, see e.g., Zwietering *et al.* (1990) for a review. These models can be fitted to the data using nonlinear least squares and characteristic growth parameters can be derived from the fit. Our experience is, however, that parametric growth curves like e.g., Gompertz or logistic law do not always accurately describe cellular growth. For some data sets the application of these models can potentially lead to systematic errors, because the functional relation between time and growth is not obvious, introducing considerable alterations in the conclusions derived from growth curve experiments.

Birch (1999) introduced a generalised model of growth equations in support of the notion that knowledge of the underlying mathematical model may be not essential, but the reliable esti-

Model	Formula	Parameter
Logistic	$y(t) = \frac{A}{1+\exp\left(\frac{4\mu}{A}(\lambda - t)+2\right)}$	A, μ, λ
Gompertz	$y(t) = A \cdot \exp\left[-\exp\left(\frac{\mu \cdot e}{A}(\lambda - t) + 1\right)\right]$	A, μ, λ
modified Gompertz	$y(t) = A \cdot \exp\left[-\exp\left(\frac{\mu \cdot e}{A}(\lambda - t) + 1\right)\right] + A \cdot \exp\left(\alpha(t - t_{shift})\right)$	$A, \mu, \lambda, \alpha, t_{shift}$
Richards	$y(t) = A \cdot \left[1 + \nu \cdot \exp\left(1 + \nu + \frac{\mu}{A} \cdot (1+\nu)^{1+1/\nu} \cdot (\lambda - t)\right)\right]^{(-1/\nu)}$	A, μ, λ, ν

Table 1: Growth $y(t)$ as function of time t for the models implemented in **grofit**.

mation of the characteristic growth parameters. In accordance with this we apply model-free spline fits in addition to the conventional parametric fit to estimate characteristic parameters from the growth curve.

The methods applied for fitting growth curves and for deriving doses response curves are described in the second section. Sections 3-4 describe the functions and the application of **grofit**. The package is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=grofit` and also from the developers website `http://www.rheinahrcampus.de/Software.2447.0.html`.

2. Methods

2.1. Fitting of growth curves

grofit applies two different strategies for fitting a given growth curve: Model-based fits and model-free spline fits. The former requires a mathematical model for the description of cellular growth. Four different models (Table 1) were implemented in **grofit**: 1. Logistic growth, 2. Gompertz growth, 3. modified Gompertz growth and 4. Richards growth (Table 1). All these models have at least three characteristic parameters: the length of lag phase λ, the growth rate μ and the maximum cell growth A. The features of these parameters are illustrated in Figure 2. The modified Gompertz growth model and the Richards model offer some flexibility utilizing additional parameter values. The modified Gompertz law enables a second increase after the function enters a first saturation plateau. Here, the shifting parameter t_{shift} and the scaling factor α control the location (time) and the strength (slope) of the second increase. The shape exponent of Richards law enables flexible adjustment that the point of inflexion can be at any value between zero and A, see Zwietering *et al.* (1990) for details.

The fitting of the parametric growth models is described in the algorithm from Table 2. Nonlinear least square fits require suitable starting values for the parameter values to be estimated. Starting values are obtained from local weighted regression fit `lowess` (Cleveland 1979). The **nls** (Bates and Watts 1988; Bates and Chambers 1992) package is used for nonlinear least squares fitting of these models. Decisions pertaining which model fits the data best are drawn according to an Akaike information criterion (Akaike 1973). According to this, the best fitting model is then used to estimate the growth parameters λ, μ and A. In addition, the area under the curve is estimated by numerical integration as an alternative characteristic of cellular growth (Hasenbrink *et al.* 2006).

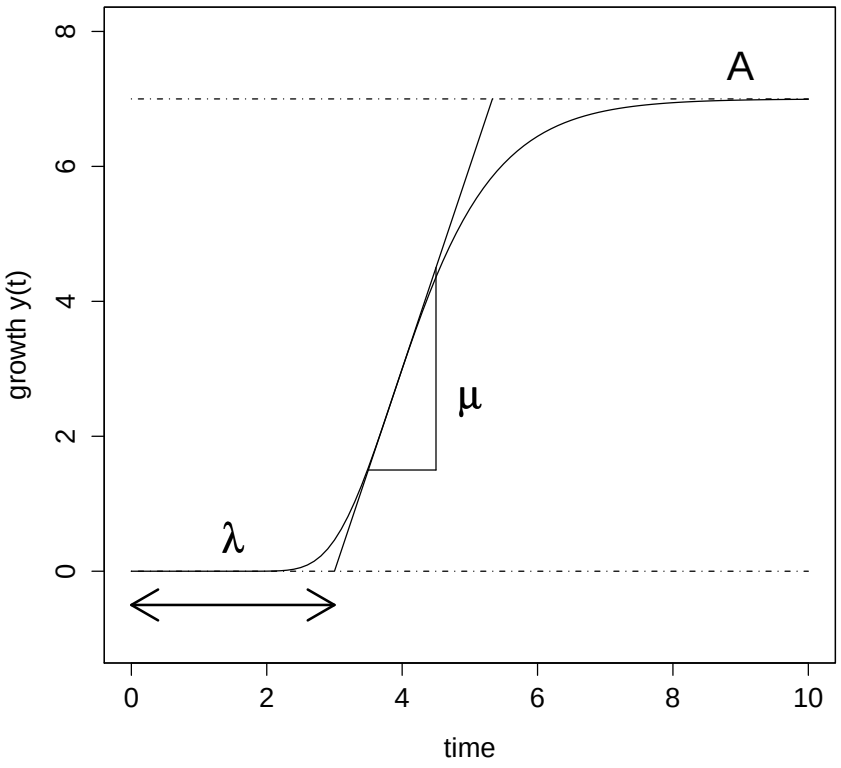

Figure 2: Typical parameters derived from growth curves: length of lag phase λ, growth rate represented by the maximum slope μ and the maximum cell growth A. The integral (area under the curve) is also used as growth parameter.

Parametric growth curves are useful and straight forward to interprete when they accurately fit the data. We noticed, however, that often real data cannot sufficiently be described by using a parametric model. As an alternative we implemented a model-free method (Table 3). This model free fit applies a smoothed cubic spline as it is implemented in the R function `smooth.spline`. A spline fit *per se* does not assume a functional relationship between time and growth data. The smoothness can individually be set by a parameter and an optimal value of this parameter can be identified by the program using cross-validation techniques.

Figure 3 shows the main differences between the two approaches. All four characteristic growth parameters (λ, μ, A and the area under the curve) were estimated from the parametric model and from the spline fit. According to the Akaike criterion, the best fitting parametric model was the logistic equation approaching the limit A for large values of time t. However, the maximum growth A was not reached by the actual data points. In this example, a diauxic shift prevents the curve from reaching saturation in the observed time interval. Thus, a more reliable parameter of growth is the maximum growth rate μ. It is estimated from the maximum slope of the fitted growth curve. It appears conclusive that the smoothing spline gives a more accurate estimate of μ. We conclude therefore that the derivation of descriptive

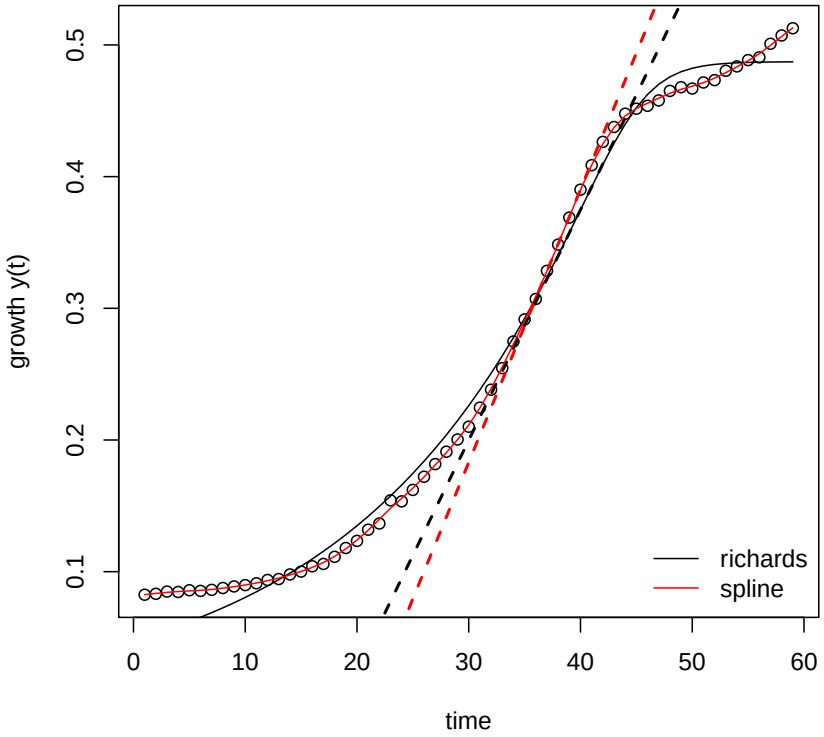

Figure 3: Comparison of parametric and model free spline fits. The growth data (circles) were fitted by a spline fit (black line). The maximum slope of the spline fit was used as an estimate for the growth rate μ. This estimate is more accurate than the best fitting parametric model (Richards equation, red lines), as can be seen from the difference in the slopes of the tangents (straight lines).

characteristics from parametric fits may potentially lead to unreliable predictions. A spline fit in contrast produces more accurate estimates of the characteristic growth parameters.

Confidence intervals for μ and the other parameters are estimated by using a nonparametric bootstrap method (Efron and Tibshirani 1993), see algorithm from Table 4. A bootstrap sample (with replacement) is generated from the original data. For each of the bootstrap samples, the characteristic parameter values λ, μ and A are estimated. These values can be displayed as bootstrap histograms (`plot`) and provide a visual guide to the variability of the different growth parameters. Standard bootstrap estimates are computed for the mean values and confidence intervals of λ, μ and A. The same procedure is also performed for the integral of the growth curve.

2.2. Fitting of dose response curves

Each of the four parameters derived from the growth curves at different compound concentrations can in principle be perceived as a characteristic response variable. The decision for one particular parameter depends in most cases on the specific experiment performed. It is advised to empirically estimate, which of the variables computed by **grofit** shows the highest sensitivity and indicates changes in growth most reliable. In addition, the confidence intervals obtained from fitting the growth curves provide further guidance to the best choice of the response variable. From our experiences with cellular growth (Hasenbrink *et al.* 2006) data we conclude that in many cases the maximum growth rate μ is a reliable descriptor.

A dose response curve was derived from a fit of the selected growth parameter *versus* the dose, see again Figure 1(c). Here we applied again the spline technique to receive a model-free relationship between dose and response (algorithm from Table 5). This offers the advantage that a large variety of different dose-response functions can be captured with a single method. From the resulting curve the EC50 value was estimated. EC50 confidence intervals were obtained by a bootstrap method (algorithm from Table 6). Bootstrap histograms for the EC50 and related parameters can be plotted for visual inspection.

3. Program application

grofit comes as a R package enabling the user to decide which functions are actually requested thus providing maximum flexibility. Nevertheless, the package can be used to run a standard workflow including all provided features.

To use **grofit** the R version 2.9.0 or later must be installed on your System. Visit `http://www.r-project.org` for downloading the latest version.

3.1. Preparation of input data

This describes the data format requested by the `grofit` function that implements the standard workflow. Data can be imported preferably from `*.csv` files (comma separated values), that are generated and read by any standard spreadsheet program. To run **grofit** the data must be arranged in a special format. The function requires a numeric matrix with time data and a `data.frame` object containing growth data as well as additional information.
Here we describe the numeric matrix named time. It consists of n columns for the time points

Table 2: Algorithm for parametric fit (`gcFitModel`)

```
Input time and corresponding growth data for different conditions
Call gcFitSpline with lowess option to estimate initial values for parametric fit
for all datasets do
  for all parametric models do
    try to fit data according to currentModel by using R-function nls
    if fit successful then
      determine AIC
      if  currentAIC < bestAIC then
        bestAIC = currentAIC
        bestModel = currentModel
      end if
    end if
  end for
end for
Determine characteristic values (A, μ, λ, integral) for bestModel
```

Table 3: Algorithm for model free fit (`gcFitSpline`)

```
Input time and related growth data
Call R-function smooth.spline
Estimate characteristic growth parameters from spline fit
Call R-function lowess
Estimate characteristic growth parameters from local weighted average fit
```

Table 4: Algorithm for parametric fit (`gcFitModel`)

```
Input time and related growth data of size k
for 1:number of bootstrap samples do
  Choose with replacement a random sample of length k from given data
  Call gcFitSpline with random samples of time and growth
  Store characteristic growth values (A, μ, λ, integral)
end for
Generate bootstrap mean and confidence intervals for (A, μ, λ, integral)
```

Table 5: Algorithm for dose response curve estimation (`drFitSpline`)

Input concentration and respecting characteristic growth parameter
Call R-function `smooth.spline` to estimate dose response curve
Estimate EC50 from spline fit

Table 6: Algorithm for bootstrap of dose response curve (`drBootSpline`)

Input concentration and respecting characteristic growth parameter
for 1:number of bootstrap samples **do**
  Call R-function `smooth.spline` to estimate dose response curve
  Estimate EC50 from spline fit
  Store EC50
end for
Generate bootstrap mean and confidence intervals for EC50

and m rows for the different experiments.

$\ddots$	t_1	t_2	...	t_n
Experiment 1	1	2	...	8
Experiment 2	1	3	...	9
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Experiment m	2	3	...	7

(1)

Note: the matrix only contains the numerical values and not the description of the rows and columns. In practical use the user may work with standard time points therefore having only a vector of time data. To construct a matrix of time points just type

```
timepoints <- 1:15
```

```
time <- t(matrix(rep(timepoints, m), c(n, m)))
```

where n refers to the number of time points and m to the number of data sets.
Next we describe a `data.frame` object, named `data`. A data.frame is nothing but a matrix except the fact that it may contain different data types. Requested is a `data.frame` of m rows belonging to each experiment and $3 + n$ columns containing growth data corresponding

to `time` and additional information. Therefore the data.frame appears as:

$\ddots$	Experiment Id	Add. info	Concentration	d_1	d_2	...	d_n
Exp 1	test_1	medium	0.13	1	2	...	20
Exp 2	test_1	high	0.23	3	5	...	19
Exp 3	test_1	medium	0.46	2	3	...	17
Exp 4	test_1	medium	0.57	1	3	...	14
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Exp m-1	test_2	low	0.12	1	3	...	23
Exp m	test_2	high	0.24	2	3	...	20

(2)

The first three columns serve to identify individual experiments. In the second column the user is free to put in any information considered as suitable. The first and the third column are the most important. In the first column one should give *one* name for an experiment that is made under a certain condition. In the third column one specifies that condition by giving a concentration of the tested substance. That means: the entry of the first column will be the same over several rows, while the entry in the third column will change. Such arragement is necessary to determine the dose response curves. It does not matter in which order the experiment ID or the concentrations appear.

Note: If one is only interested in growth curves the first three columns can be utilized in any way. But the output of the growth curve fit cannot be further used for dose-response curve fitting without manipulation.

Note: Do not use any special symbols like `/`, `*`, `ü`, `' '` (blank) etc. for description in the first three rows. These symbols may cause errors in current or future versions of R. Use `_`, `.`, or `-` instead.

Create input data using spreadsheet programs

An alternative to data import and manipulation is the use of a spreadsheet program like `Excel`. Copy and paste the data in the format as it was described above. In case of missing values the according cell is left empty. Then export the sheet as a csv-file (or `*.txt` either). In R type

```
data <- read.table("PATH/data.csv", header = FALSE, sep = ";", dec = ".")
time <- read.table("PATH/time.csv", header = FALSE, sep = ";", dec = ".")
```

Note: Depending on the language settings of your system, one may need to use other options for `sep` and `dec`. Open the exported `*.csv` file in an appropriate editor (`WordPad`, `Kate` etc.) and see documentation of `read.table` to find out the necessary options.

3.2. Simulating data

We deposited a dataset for testing purposes in the data folder of the package. To create simulated data sets we included the function `ran.data`. To generate a data set with 30 timepoints and 40 datasets type:

```
dataset <- ran.data(40, 30, 1, 5, 15)
```

```
data <- dataset$data

time <- dataset$time
```

This will generate data according to Gompertz law using $\mu = 1$, $\lambda = 5$ and $A = 15$. These parameters are slightly changed in each data set to simulate the dependence of growth curves from a substance. In addition some noise is added to the generated data

3.3. Options

The options can be set by the call of the `grofit.control` function, which returns an object of class `grofit.control`, depicting a list including all **grofit** options. To change the default values type `MyOpt <- grofit.control(fit.opt = "s", model.type = c("gompertz"), ...)` The options are related to different parts of the program and are described in detail below.

3.4. Common options

- `neg.nan.act`: logical (`TRUE`/ `FALSE`), indicates whether the program should stop when negative growth values or non numerical values appear (`TRUE`). Otherwise the program remove these values silently (`FALSE`). Improper values may be caused by incorrect data or input errors. Default: `FALSE`.

- `clean.bootstrap`: logical, determines if negative values which occur during bootstrap should be removed (`TRUE`) or kept (`FALSE`). Note: Infinite values were always removed. Default: `TRUE`.

- `suppress.messages`: logical, determines if **grofit** messages like information about current growth curve, EC50 values etc. shall be printed to screen (`FALSE`) or not (`TRUE`). This option serves to speed up the processing of high troughput data. Note: warning messages are still displayed. Default: `FALSE`.

3.5. Fit growth curve options

- `fit.opt`: indicates whether the program should perform a model fit `"m"`, a spline fit `"s"` or both `"b"`. Default: `"b"`.

- `log.x.gc, log.y.gc`: logical, indicates whether a $\ln(x + 1)$. transformation should be applied to time (x-axes) or growth values (y-axes). Default: `FALSE`.

- `interactive`: logical, controls whether the fit of each growth curve is controlled manually by the user. Default: `TRUE`.

- `nboot.gc`: number of bootstrap samples used for the model free growth curve fitting. Use `0` to disable the bootstrap. Default: `0`.

- `smooth.gc`: parameter describing the smoothness of the spline fit; usually (not necessary) $\in (0; 1]$. Set `smooth.gc` to `NULL` causes the program to query an optimal value via cross validation techniques. Note: This is partly experimental. In future improved implementations of the `smooth.spline` function may lead to different results. See documentation of the R function `smooth.spline` for further details. Especially for datasets with few data points the option `NULL` might result in a too small smoothing parameter, which produces an error in `smooth.spline`. In that case the usage of a fixed value is recommended. Default: `NULL`.
- `model.type`: string vector, giving the names of the parametric models which should be fitted to the data. The addition of user defined models is described in Section 4.4. Default: `c("logistic", "gompertz", "richards", "gompertz.exp")`.

3.6. EC50 options

- `have.atleast`: minimum number of different values for the growth parameter one should have for estimating a dose-response curve. Note that the bootstrapping procedure needs at least six values. Default: `6`.
- `parameter`: The column in the output table which should be used for creating a dose response curve. See the description of the output table in Section 4. Usually you will use numbers $\in [9; 12]$ and $\in [28; 35]$. Default: `9` (which represents μ from the parametric fit).
- `smooth.dr`: parameter describing the smoothness of the spline fit; usually (not necessary) $\in (0; 1]$. See documentation of the R function `smooth.spline` for further details. Default: `NULL`.
- `log.x.dr, log.y.dr`: logical, indicates whether a $\ln(x + 1)$ transformation should be applied to dose (x-axes) or response (y-value). Default: `FALSE`.
- `nboot.dr`: number of bootstrap samples for the EC50. Use `0` to disable bootstrapping. Default: `0`.

4. Program run

4.1. Standard workflow

Having the data successfully arranged one types

```
TestRun <- grofit(time, data, TRUE)
```

to run the program.

This will run the standard workflow of the program with the default options specified in `grofit.control`. It is separated into two major parts: the growth curve fitting and the dose

response curve fitting (see Figure 4). First, the function `gcFit` is called to perform the curve fit with the desired options. This procedure includes the parametric fit, the model free fit and respective bootstrapping.

Note: One should not get confused by the error messages, which will occur during the call of the function `gcFitModel`. These messages result from the R procedure `nls` and indicate that a certain parametric model could not be fitted to data. This inidicates that different models are required.

The result of `gcFit` is an object of class `gcFit`. See documentation for further details.

The output table of `summary(gcFit)` serves as the input for `drFit`. The function autonomously reads the table and determines the number of different experiments by using the experiment ID in the first column of the table. Those which appear to have less valid values than specified by `have.atleast` will be automatically removed.
If one is only interested in growth curve fitting type:

```
TestRun <- grofit(time, data, FALSE)
```

This will only run the `gcFit` function. Advanced users may use the given functions apart from the standard workflow. We separated the program in modular parts that can be called individually. See documentation for further details.

4.2. Example

In this section we provide an example for the standard workflow of **grofit**. The example corresponds to Figure 1 and describes the fitting of growth curves (Figure 1 a) by using the function `gcFit` and the subsequent estimation of the corresponding dose response curve (Figure 1 b) and c) by `drFit`. One should carefully compare the workflow in the flowchart in Figure 4 to monitor the different processing steps and have also a look to the options and default values described in Section 3.3.

1. In a first step appropriate options can be set by the function `grofit.conctrol`, see Section 3.3 for details. Following two slightly different settings are defined.

   ```
   MyOpt1 <- grofit.control(smooth.gc = 0.5, parameter = 28,
             interactive = FALSE)
   MyOpt2 <- grofit.control(smooth.gc = 0.5, parameter = 28,
             interactive = FALSE, log.x.dr = TRUE)
   ```

   This sets the smoothness of the spline fit of growth curves (`smooth.gc`), chooses μ of the model free fit as response parameter (`parameter`) and disables the interactive mode. Differing from the first option, `MyOpt2` enables the logarithmic transformation of the concentrations for the dose response curve (`log.x.dr`).

2. The example data is part of the **grofit** package and is stored in the variables `grofit.data` and `grofit.time`.

3. To start the standard workflow with the one and the other option type:

   ```
   TestRun1 <- grofit(grofit.time, grofit.data, TRUE, MyOpt1)
   TestRun2 <- grofit(grofit.time, grofit.data, TRUE, MyOpt2)
   ```

The parameter `TRUE` indicates that a dose response curve will be estimated from the growth curves.

In both cases `grofit` does the following: first the `grofit` function calls the `gcFit` function that performs the growth curve fitting. The option `fit.opt = "b"` (*b*oth, default value; see Section 3.5) means that both the parametric and the model free fit will be performed.

The parametric fit is processed by the `gcFitModel` function that utilises the R internal function `nls`. A guess of initial values for A, μ, λ is obtained from a local weighted regression method (R function `lowess`) which is calculated by the `gcFitSpline` function. These values are passed on to the `initMODEL` function generating initial values for possible additional parameters (in case of Richards or modified Gompertz law). Due to the option `model.type = c("logistic", "gompertz", "richards", "gompertz.exp")` the program tries to fit every available parametric model. On the screen the status of the fit is shown: `OK`, `nls() failed to converge with stopCode` or `ERROR in nls()`. If `nls` fails to converge the respecting `stopCode` (see documentation of `nls`) is provided. The `ERROR` status usually results from sigular gradients or infinite values produced during the call of the `nls` function. This quite frequent error is not to be taken critical and indicates only that a certain model is not an appropriate description of the growth curve (see also documentation of `gcFitModel`).

Then, `gcFit` calls the `gcFitSpline` function to perform a model free spline fit by using the R internal function `smooth.spline`.

In the interactive mode (not in this example) `grofit` presents both data fits and asks the user for consistence with expectations. If the fit satisfies your expectations chose `y`, otherwise the growth curve will be excluded from further analysis.

Following fitting the last growth curve, the `gcFit` function returns the calculated growth parameters as an object of class `gcFit` to the `grofit` function.

The output of `summary(gcFit)` serves then as an input of `drFit` to generate dose response curves. In the workspace a short message informs about the number of different experiments and the number of valid datasets per experiment. Here the example dataset pertains to one experiment with parameters from seven growth curves. In the workspace the half maximal effective concentration (EC50) and the corresponding response value are shown. The following lines gives the output produced by `drFit` in case of `TestRun1`.

```
=== EC 50 Estimation ============================
-------------------------------------------------
--> Checking data ...
--> Number of distinct tests found: 1
--> Valid datasets per test:
      TestID Number
      Test1  7


=== Dose response curve estimation ===============
--- EC 50 ---------------------------------------
--> Test1
xEC50 6.61638638638639 yEC50 0.0208673971407452
```

The complete code to reproduce Figures 1(a), 1(b) and 1(c):

Define options

```
MyOpt1 <- grofit.control(smooth.gc = 0.5, parameter = 28,
           interactive = FALSE)
MyOpt2 <- grofit.control(smooth.gc = 0.5, parameter = 28,
           interactive = FALSE, log.x.dr = TRUE)
```

Run grofit

```
TestRun1 <- grofit(grofit.time, grofit.data, TRUE, MyOpt1)
TestRun2 <- grofit(grofit.time, grofit.data, TRUE, MyOpt2)
```

Defining color and plot symbol vector

```
colData <- c("black", "cyan", "magenta", "green", "blue", "orange", "grey")
pch     <- 1:7
dev.new(width = 8, height = 3)
par(mfrow = c(1, 3), mar = c(5.1, 4.1, 3.1, 1.1))
```

Generate Fig. 1a, b, c

```
plot(TestRun1$gcFit, opt = "s", colData = colData, colSpline = 1,
       pch = pch, cex = 1)
title("a")
plot(TestRun1$drFit$drFittedSplines[[1]], colData = colData,
       pch = pch, cex = 1)
title("b")
plot(TestRun2$drFit$drFittedSplines[[1]], colData = colData,
       pch = pch, cex = 1)
title("c")
```

Another useful example to test the effect of `log.x.dr`, is to choose `parameter = 9` (maximum slope of the parametric fit) for the dose response curve. While `log.x.ec = FALSE` leads to an unreliable dose response curve, `log.x.dr = TRUE` produces acceptable results.

It is also recommended to try out the different generic `plot` functions for objects of class `gcFit`, `gcBootSpline`, `gcFitModel`, `drFitSpline` and `drBootSpline`.

Logarithmic transformation might be useful in cases when the data points are not equally distributed over the x-axes. However, data fitting is always a delicate issue so that we can not provide general recommendations for data transformations or the choice of certain smoothing parameters.

4.3. Performance

The performance was tested on an IBM T43 Notebook (Intel Pentium M 2GHz processor with 1GB RAM and Windows XP Servicepack2) using R version 2.2.1. For a fair comparison we used the automatic mode. For 100 growth curves each comprising 25 data points the parametric fit, the model free fit and dose response curve estimation, took in total 19 sec. Enabling bootstrap samples for the model free fit, as well as for the dose response curve (100

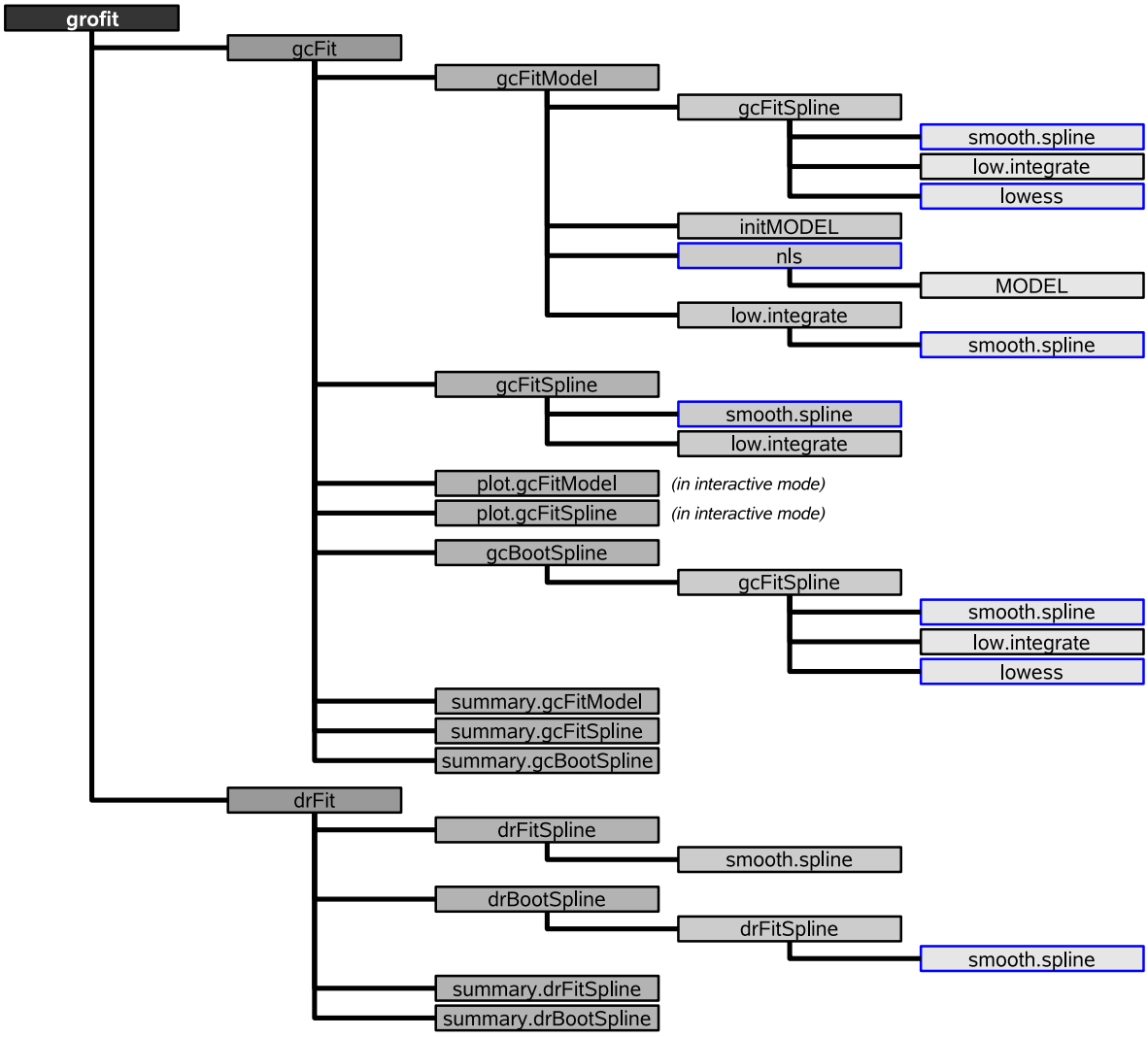

Figure 4: The flowchart shows an overview of the standard workflow. **grofit** contains two major components: `gcFit` and `drFit`. `gcFit` executes the routines for parametric (`gcFitModel`) and model free (`gcFitSpline`) growth curve fitting as well as a bootstrap procedure (`gcBootSpline`) for the model free fit. `gcFitModel` depends on several functions of the **grofit** package and also on the R internal function `nls`. `gcFitSpline` provides the model free spline fit and `gcBootSpline` creates a respective bootstrap sample by conducting several times `gcFitSpline`. `drFit` uses the output of `gcFit` to relate concentrations to certain characteristic growth values. The dose response curve fit and EC50 estimation is performed by `drFitSpline` (using R function `smooth.spline`), whereas statistics of EC50 estimation are obtained from a bootstrap sample given by `drBootSpline`. Blue boxes indicate R internal functions.

bootstrap samples) took 1 min. 51 sec. It appears thus reasonable to assume that the package is also applicable to high throughput datasets.

4.4. Adding parametric models

The user can implement his own parametric growth model by writing a model definition file and a function to generate respective initial values for the parameter estimation. To create a model file, type `fix(NEWMODEL)`. As a common standard in **grofit** the model definition has to be dependent on time, A, μ, λ and a numeric vector that contains additional parameters, used e.g., for Richards or the modified Gompertz growth law (see Table 1). In case that no additional parameters are necessary initialize `addpar = NULL` in the function header.

Example for model definition without additional parameters

```
NEWMODEL <- function (time, A, mu, lambda, addpar = NULL)
{
        NEWMODEL <- A / (1 + exp(4 * mu * (lambda - time) / A + 2))
}
```

Example for model definition with additional parameters

```
NEWMODEL <- function (time, A, mu, lambda, addpar)
{
alfa   <- addpar[1]
tshift <- addpar[2]
e      <- exp(1)
y      <- A * exp(-exp(mu * e * (lambda - time) / A + 1))
                                + A * exp(alfa * (time - tshift))
        NEWMODEL <- y
}
```

The model definition file should be saved as `NEWMODEL.R`

The function to generate respective initial values follows the name convention `initNEWMODEL`. It must be dependent on time, growth data, A, μ and λ. These parameters can be used to calculate the initial values, which will be used in the R function `nls` during the run of `gcFitModel`. The function returns a list object comprising all parameters that are implemented in the model definition. One should ensure to initialize `addpar = NULL` in case that no additional parameters are used.

Example for initial value function for a model without additional parameters

```
initNEWMODEL <- function (time, y, A, mu, lambda)
{
A      <- max(y)
mu     <- mu
lambda <- lambda
initNEWMODEL <- list (A = A, mu = mu, lambda = lambda, addpar = NULL)
}
```

Example for initial values function for a model with additional parameters

```
initNEWMODEL <- function (time, y, A, mu, lambda)
{
alfa   <- 0.1
tshift <- max(time) / 10
A      <- max(y)
mu     <- mu
lambda <- lambda
initNEWMODEL <- list(A = A, mu = mu, lambda = lambda,
                                     addpar = c(alfa, tshift))
}
```

If one creates the functions with an editor outside the R environment, use of the `source` command should be applied to make the functions available to R. To implement the new model the string `"NEWMODEL"` in the `model.type` option (see Section 3.5) must be added. During the program run **grofit** will search for the functions `NEWMODEL` and `initNEWMODEL` and stop automatically with an error message if any of these is missing or not in the correct format. Most likely such error messages are caused by simple spelling mistakes or due to the lack of the correct `source` command.

grofit allows for adjusting specific settings like type of fit, logarithmic data transformation or the number of bootstrap samples. The growth curve fit can be processed automatically or in an interactive mode. In the interactive mode the user is allowed to exclude unreliable measurements from dose response curve estimation.

To generate graphical out the package provides generic plot functions for objects of the classes `gcFit`, `drFit`, `gcFitModel`, `gcFitSpline`, `gcBootSpline`, `drFitSpline` and `drBootSpline` created by the functions of the same name.

The two output tables of presumably major interest are generated by the generic `summary` functions for `drFit` and `gcFit` and described in the following tables labeled *gcFit* and *drFit* (Tables 7, 8, 9, 10, 11).

5. Discussion and Conclusions

grofit is a useful tool for all scientists that employ biological growth analysis. Its properties reduce the potential systematic error by enabling the user to carefully control the type of fitting. It is anticipated that this type of control will correspondingly produce quite reliable results. An earlier version of the package was used for Hasenbrink *et al.* (2006) and is therefore assumed to be rather well tested.

Growth curve modeling is also popular in areas outside biology. For example, in economic theory is much interest in relating government expenditures to economic growth. In the social sciences, the analysis of growth curves encounters a fairly long tradition. Typically, observations made on many individuals across pretest and post-test occasions are compared and the effect of several covariables is analysed. In many cases, a parametric model for growth is assumed or the inference is based on a general linear model Duncan *et al.* (2006). While these applications may share some similarity with the specific application described above there are some special issues that are not implemented in **grofit**.

The typical application of **grofit** considers a situation where a biomass or a similar quantitative

gcFit

Column Number	Column name	Description
1	`test.id`	Name of the experiment
2	`add.id`	Additional information
3	`concentration`	Concentration of substrate
4	`reliability`	reliability flag
5	`use.model`	parametric model used
6	`log.x`	logarithmic transformation
7	`log.y`	logarithmic transformation
8	`nboot.fit`	number of bootstrap samples

Table 7: Description of *gcFit*. Each row of the above table is generated in the `gcFit` function.

gcFit continued

Column Number	Column name	Description
9	`mu.model`	max. slope μ
10	`lambda.model`	lag-phase λ
11	`A.para`	maximum growth
12	`Integral.model`	integral
13	`stdmu.model`	standard deviation μ (cross validation)
14	`stdlambda.model`	standard deviation λ (cross validation)
15	`stdA.model`	standard deviation A (cross validation)
16	`ci90.mu.model.lo`	90 % CI lower boundary μ
17	`ci90.mu.model.up`	90 % CI interval upper boundary μ
18	`ci90.lambda.model.lo`	90 % CI interval lower boundary λ
19	`ci90.lambda.model.up`	90 % CI interval upper boundary λ
20	`ci90.A.model.lo`	90 % CI interval lower boundary A
21	`ci90.A.model.up`	90 % CI interval upper boundary A
22	`ci95.mu.model.lo`	95 % CI interval lower boundary μ
23	`ci95.mu.model.up`	95 % CI interval upper boundary μ
24	`ci95.lambda.model.lo`	95 % CI interval lower boundary λ
25	`ci95.lambda.model.up`	95 % CI interval upper boundary λ
26	`ci95.A.model.lo`	95 % CI interval lower boundary A
27	`ci95.A.model.up`	95 % CI interval upper boundary A

Table 8: Description of *gcFit*. Each row of the above table is generated by the call of generic `summary` function for `gcFitModel` objects.

gcFit continued

Column Number	Column name	Description
28	`mu.spline`	max. slope μ
29	`lambda.spline`	lag-phase λ
30	`A.nonpara`	maximum growth
31	`integral.spline`	integral

Table 9: Description of *gcFit*. Columns 28-31 are generated by the call of the function generic `summary` function for `gcFitSpline` objects.

gcFit continued

Column Number	Column name	Description
32	`mu.bt`	mean of bootstrap μ
33	`lambda.bt`	mean of bootstrapλ
34	`A.bt`	mean of bootstrap$A$
35	`integral.bt`	mean of bootstrap integral
36	`stdmu.bt`	standard deviation μ (bootstrap)
37	`stdlambda.bt`	standard deviation λ (bootstrap)
38	`stdA.bt`	standard deviation A (bootstrap)
39	`stdIntegral.bt`	standard deviation integral (bootstrap)
40	`ci90.mu.bt.lo`	90 % CI lower boundary μ
41	`ci90.mu.bt.up`	90 % CI upper boundary μ
42	`ci90.bt.lambda.lo`	90 % CI lower boundary λ
43	`ci90.bt.lambda.up`	90 % CI upper boundary λ
44	`ci90.A.bt.lo`	90 % CI lower boundary A
45	`ci90.A.bt.up`	90 % CI upper boundary A
46	`ci90.integral.bt.lo`	90 % CI lower boundary integral
47	`ci90.integral.bt.up`	90 % CI upper boundary integral
48	`ci95.mu.bt.lo`	95 % CI lower boundary μ
49	`ci95.mu.bt.up`	95 % CI upper boundary μ
50	`ci95.lambda.bt.lo`	95 % CI lower boundary λ
51	`ci95.lambda.bt.up`	95 % CI upper boundary λ
52	`ci95.A.bt.lo`	95 % CI lower boundary A
53	`ci95.A.bt.up`	95 % CI upper boundary A
54	`ci95.integral.bt.lo`	95 % CI lower boundary integral
55	`ci95.integral.bt.up`	95 % CI upper boundary integral

Table 10: Description of *gcFit*. Columns 32-55 are generated by the call of the generic `summary` function for `gcBootSpline` objects.

drFit

Column Number	Column name	Description
1	`name`	name of experiment
2	`log.x`	logarithmic transformation
3	`log.y`	logarithmic transformation
4	`Samples`	number of bootstrap samples
5	`EC50`	EC50 value
6	`yEC50`	y value corresponding to EC50
7	`EC50.orig`	EC50 value in original scale
8	`yEC50.orig`	y value EC50 in original scale
9	`meanEC50`	mean EC50 from bootstrap
10	`sdEC50`	standard deviation EC50 (bootstrap)
11	`ci90EC50.lo`	90 % CI lower boundary (bootstrap)
12	`ci90EC50.up`	90 % CI upper boundary (bootstrap)
13	`ci95EC50.lo`	95 % CI lower boundary (bootstrap)
14	`ci95EC50.up`	95 % CI upper boundary (bootstrap)
15	`meanEC50.orig`	mean EC50 from bootstrap in original scale
16	`ci90EC50.orig.lo`	90 % CI lower boundary in original scale
17	`ci90EC50.orig.up`	90 % CI upper boundary in original scale
18	`ci95EC50.orig.lo`	95 % CI lower boundary in original scale
19	`ci95EC50.orig.up`	95 % CI upper boundary in original scale

Table 11: Description of output (*drFit*). Each row of the above table is generated in the `drFit` function. Description of *drFit* continued. Columns 5-8 are generated by the call of the generic `summary` function for `drFitSpline` objects. Columns 9-19 are generated by the call of the generic `summary` function for `drBootSpline` objetcs.

variable is measured over time under different experimental conditions. In this paper, we used the example of cellular growth in the presence of different concentrations of a chemical compound. Similar situations appear in other areas of biology and medicine. For example, a researcher studying the effect of a certain drug on obesity will measure the animal weight over time for different drug doses. In some cases, the growth curves can be described parametrically in other situations the implemented model free approach based on splines might be more appropriate. **grofit** offers the flexibility to derive characteristic growth parameters like A, μ, λ from parametric and model free fits. The effect of quantitative variables like compound or drug doses can correspondingly be studied by estimating a dose response relationship. Typically, such data exhibit a much higher degree of scatter around the estimated ideal growth curve than our example of cellular growth. This variability is *per se* taken into account by the implemented bootstrap technique. The bootstrap confidence intervals for the characteristic growth parameters will in such cases typically be larger reflecting the lower precision of the measurements.

The related R package **agce** (Gottardo 2006) provides methods to compare growth curves under different conditions using MANOVA and similar methods. The **agce** code is indeed useful when the growth curve is a linear function of time or when the data can be transformed to be approximately linear. **grofit** can be used for nonlinear growth curves where linear statistical methods can not be applied. Another useful package depicts **drc** by Ritz and Streibig (2005). The authors focus on the parametric fit of dose response curves and also provide access for statistical analysis. Although **grofit** allows in principle the application of the parametric fit routine `gcFitModel` to dose-response curve data, **drc** is much more specialized and thereore recommended to users with a certain interest in parametric dose response curves.

In the current version of **grofit** the effect of a covariable like e.g., drug concentration can be analyzed by comparison of one of the characteristic parameters A, μ or λ with different values of the covariable. The advantage of this approach is that standard univariate statistical techniques can be applied. For example, if growth curves for a control and a treatment situation are to be compared one can use a two sample statistical test to detect significant effects of the treatment. The disadvantage of this approach is however, that the characteristic parameter values only display a certain effect on the growth curve and can not in general capture the treatment effect on the growth curve as a whole. We intend to improve **grofit** to compute the influence of covariates in a more general way, see e.g., Altmann and Casella (1995).

Acknowledgment

The authors thank Manfred Berres for helpful discussions. This work was supported by the Federal Ministry of Education and Research (BMBF grant 031 3982 C) as part of the TRANSLUCENT project within the SysMO (Systems Biology of Microorganisms) initiative.

References

Airoldi E, Huttenhower C, Gresham D, Lu C, Caudy A (2009). "Predicting Cellular Growth from Gene Expression Signatures." *PLoS Computational Biology*, **5**(1), e1000257.

Akaike H (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In BN Petrov, F Csaki (eds.), *Second International Symposium on Information Theory*, pp. 267–281. Akademiai Kiado.

Alarcorn T, Tindall M (2007). "Modeling Cell Growth and its Modulation of the G1/S Transition." *Bulletin of Mathematical Biology*, **69**, 197–214.

Altmann N, Casella G (1995). "Nonparametric Empirical Bayes Growth Curve Analysis." *Journal of the American Statistical Association*, **90**, 508–515.

Bates DM, Chambers JM (1992). *Nonlinear Models*. Chapman & Hall, London.

Bates M, Watts G (1988). *Nonlinear Regression Analysis and its Application*. John Wiley & Sons, New York.

Birch C (1999). "A New Generalized Logistic Sigmoid Growth Equation Compared with the Richards Growth Equation." *Annals of Botany*, **83**, 713–723.

Cleveland W (1979). "Robust Locally Weighted Regression and Smoothing Scatterplots." *Journal of American Statistical Association*, **74**, 829–836.

Duncan T, Duncan S, Strycker L (2006). *An Introduction to Latent Variable Growth Curve Modeling: Concepts, Issues, and Applications*. Lawrence Erlbaum Associates Inc, Mahwah, NJ.

Efron B, Tibshirani RJ (1993). *An Introduction of the Bootstrap*. Chapman & Hall, London.

Fath B, Jorgensen S, Patten B, Straskraba M (2004). "Ecosystem Growth and Development." *BioSystems*, **77**, 213–228.

Gottardo R (2006). ***agce**: Analysis of Growth Curve Experiments*. R package version 1.2, URL http://CRAN.R-project.org/package=agce.

Hasenbrink G, Kolacna L, Ludwig J, Sychrova H, Kschischo M, Lichtenberg-Fraté H (2006). "Ring Test Assessment of the mKir2.1 Growth Based Assay in Saccharomyces cerevisiae Using Parametric Models and Model-free Fits." *Applied and Microbiology Biotechnology*, **73**, 1212–1221.

Jorgensen S, Patten B, Straskraba M (2000). "Ecosystems Emerging: 4. Growth." *Ecological Modeling*, **126**(2-3), 249–284.

Kahm M, Hasenbrink G, Lichtenberg-Fraté H, Ludwig J, Kschischo M (2010). "**grofit**: Fitting Biological Growth Curves with R." *Journal of Statistical Software*, **33**(7), 1–21. URL http://www.jstatsoft.org/v33/i07/.

Qu Z, Weiss J, MacLellan W (2004). "Coordination of Cell Growth and Cell Division: a Mathematical Modeling Study." *Journal of Cell Science*, **117**, 4199–4207.

Ritz C, Streibig JC (2005). "Bioassay Analysis using R." *Journal of Statistical Software*, **12**(5), 1–22. URL http://www.jstatsoft.org/v12/i05/.

Zwietering M, Jongenburger I, Rombouts F, van 'T Riet K (1990). "Modeling of the Bacterial Growth Curve." *Applied and Environmental Microbiology*, **56**, 1875–1881.

Affiliation:

Matthias Kahm
University of Applied Sciences Koblenz, RheinAhrCampus
Department of Mathematics and Technology
Südallee 2
D-53424 Remagen/Germany
E-mail: sysbio_software@rheinahrcampus.de

Guido Hasenbrink
University of Bonn
Institute for Cellular and Molecular Botany
Kirschallee 1
D-53115 Bonn/Germany

Hella Lichtenberg-Fraté
University of Bonn
Institute for Cellular and Molecular Botany
Kirschallee 1
D-53115 Bonn/Germany

Jost Ludwig
University of Bonn
Institute for Cellular and Molecular Botany
Kirschallee 1
D-53115 Bonn/Germany

Maik Kschischo
University of Applied Sciences Koblenz, RheinAhrCampus
Department of Mathematics and Technology
Südallee 2
D-53424 Remagen/Germany
E-mail: sysbio_software@rheinahrcampus.de