inactivation: Software for modeling of microbial inactivation

Alberto Garre, Jose A. Egea, Pablo S. Fernandez

2015-10-16

XX Hacer las gráficas en ggplot?? Eso implicaría añadir otra dependencia…

Introduction

R-package **inactivation** implements functions useful for the modelization of microbial inactivation. The package contains functions for making predictions for some model parameters given as well as for adjusting models to experimental data collected in the laboratory. All the functions implemented in the **inactivation** package are written in R.

Once installed, the **inactivation** package can be loaded by writing,

```
library(microinactivation)
```

inactivation uses the packages `deSolve` (Soetaert, Petzoldt, and Setzer 2010) and `FME` (Soetaert, Petzoldt, and Setzer 2010) to both predict results and adjust common inactivation models to experiment data. The inactivation models most commonly used in the industry have been implemented in this package:

- Bigelow model **XXreference**
- Weibull-Mafart model (Mafart et al, 2002),
- Weibull-Peleg model **XXreference**
- Geeraerd model (Geeraerd et al, 2000).

The **inactivation** package provides several functions which help in the modelization of microbial inactivation process:

- `predict_inactivation()` returns the predicted inactivation process, given a set of model parameters.
- `fit_isothermal_inactivation` adjusts the parameters of a selected inactivation model to a set of isothermal experimental data.
- `fit_dynamic_inactivation` fits the parameters of a given inactivation model to a set of dynamic inactivation experiments.

Moreover, several functions with miscellaneous information about the models implemented are provided.

In this document, a survey of the functionality of the **inactivation** package is provided. This demonstration is based on two example data sets included in the package itself. These packages imitate a collection of experimental data obtained for a set of isothermal and dynamic experiments.

Example data sets

The **inactivation** package contains two data sets that will be used within this vignette to illustrate

its capabilities.

The example data set for isothermal fit

The **inactivation** package includes the `isothermal_example` data set, which imitates the data obtained during a set of isothermal inactivation experiments. This data set can be loaded as:

```
data(isothermal_example)
```

This example data set has 3 variables and 36 observations.

- The first column (`time`) provides the time at which the measurement was taken,
- the second column (`temp`) gives the temperature of the experiment,
- the logarithmic difference is given by the third column (`log_diff`).

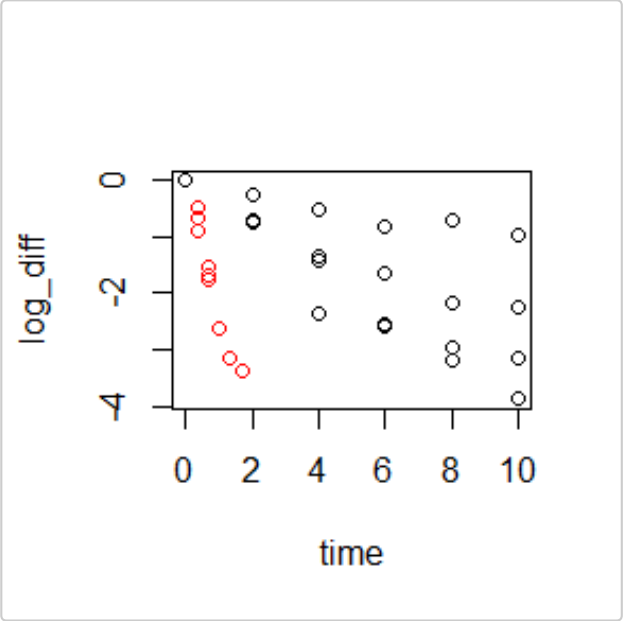
```
head(isothermal_example)
```

```
##        time temp   log_diff
## 1 0.0000000  105  0.0000000
## 2 0.3333333  105 -0.6777807
## 3 0.6666667  105 -1.6777807
## 4 0.0000000  105  0.0000000
## 5 0.3333333  105 -0.8822398
## 6 0.6666667  105 -1.5300573
```

```
summary(isothermal_example)
```

```
##       time              temp           log_diff
##  Min.   : 0.0000   Min.   : 95.00   Min.   :-3.8808
##  1st Qu.: 0.3333   1st Qu.: 95.00   1st Qu.:-2.4170
##  Median : 2.0000   Median : 95.00   Median :-1.1540
##  Mean   : 3.5278   Mean   : 98.33   Mean   :-1.4197
##  3rd Qu.: 6.0000   3rd Qu.:105.00   3rd Qu.:-0.5128
##  Max.   :10.0000   Max.   :105.00   Max.   : 0.0000
```

```
plot(log_diff  ~ time, data=isothermal_example, col = as.factor(temp))
```



XX Añadir leyenda

The example data set for dynamic fit

XX Añadir la temperatura de cada observación al data set??

The `inactivation_example` data set contained within the **inactivation** package provides a `data.set` imitating the data obtained during a non-isothermal inactivation experiment. This data set can be loaded as follows:

```
data(inactivation_example)
```

This example data set contains 2 variables and 19 observations. The first columns, `time`, reflects the time point at which the the observation was taken. The second one, `N` provides the number of microorganisms counted on the observation.

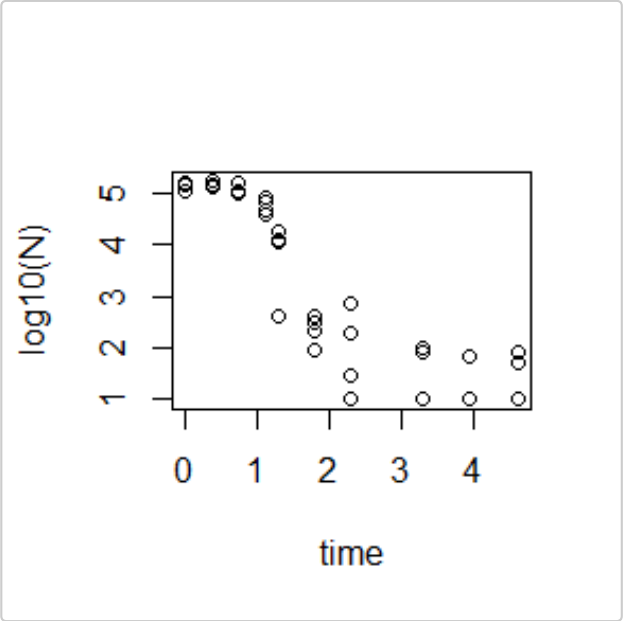
```
head(inactivation_example)
```

```
##   time      N
## 1 0.00 106000
## 2 0.37 129000
## 3 0.74 109000
## 4 1.11  68000
## 5 1.30  18000
## 6 1.80    400
```

```
summary(inactivation_example)
```

```
##       time             N
##  Min.   :0.000   Min.   :    10
##  1st Qu.:0.740   1st Qu.:    70
##  Median :1.550   Median :   580
##  Mean   :1.951   Mean   : 46178
##  3rd Qu.:3.300   3rd Qu.:102250
##  Max.   :4.630   Max.   :165000
```

```
plot(log10(N)  ~ time, data=inactivation_example)
```



XX Añadir leyenda XX Cambiar nombre del data set

Inactivation models implemented

The most commonly used models for the modelization of microbial inactivation have been implemented in the **inactivation** package: Bigelow, Weibull-Mafart, Weibull-Pelec and Geeraerd. A brief mathematical description of each one of them is presented through this section.

Bigelow's model

Bigelow's model **XXreference** assummes that the inactivation process follows a first order kinetics and, therefore, a negative linear relation exists between the logarithm of the number of microorganisms (N) and the time (t).

$$ \mathrm{log}_{10}(N) = -\frac{1}{D_T}t $$

The parameter D_T (also named D-value) equals the inverse of the slope of the line. Its relationship with temperature (T) is described by the equation

$$ \mathrm{log}_{10}D_T = \mathrm{log}_{10}D_R + \frac{T_R-T}{z} $$

where T_R is a reference temperature and D_R the D-value at this temperature. The parameter z is usually named z-value and provides an indication of the sensitivity of the microorganism to temperature variations.

XXdesarrollar modelo dinamico

Weibullian models

Weibullian models assume that the the inactivation process can be viewed as a failure phenomenon. According to this hypothesis, the time that each microorganism within the population can resists the environmental conditions given follows a probability distribution type Weibull. Two models of this family have been implemented: Weibull-Mafart and Weibull-Pelec.

Weibull-Mafart model

The Weibullian model presented by Mafart et al. (2002) for isothermal cases is described by the equation:

$$\mathrm{log}_{10}S(t) = - \Big(\frac{t}{\delta} \Big)^\beta$$

where $S = N/N_0$ represents the fraction of survivors and t the time. δ and β are respectively the rate and shape factors of the Weibull distribution.

In this model, β is assumed constant and δ follows and exponential relation with temperature (T)

$$\delta(T) = \frac {\delta_{ref}} {10^ {\Big(\frac{T-T_{ref}}{z} \Big)} }$$

where T_{ref} is a reference temperature, δ_{ref} is the value of δ at this reference temperature and z describes the sensitivity of the microorganism to temperature variations.

XXmodelo dinamico

Weibull-Peleg model

The Weibullian model presented by **XXreference** for isothermal cases follows the equation:

$$\mathrm{log}_{10}S(t) = - b \cdot t^n$$

where $S = N/N_0$ represents the fraction of survivors and t the time, and b and n are the two parameters of the model. Both parameters are equivalent to δ and β parameters of the Weibull-Mafart model as they define the Weibull distribution function:

$$n = \beta \quad b = \frac {1}{\delta^\beta}$$

XX a lo mejor quitar la comparación entre ambos modelos

In the Weibull-Peleg model the parameter n is assumed temperature independent, while the dependency of b with temperature is modeled as

$$b(T) = \mathrm{ln} \big\{ 1 + \mathrm{exp} \big[k(T - T_c) \big] \big\}$$

This equation states that for temperatures much lower than the critical one (T_c) $b(T) = 0$, so there is no inactivation. For temperatures close to T_c, there is and exponential growth of $b(T)$ with respect to the temperature. For temperatures much higher than the critical one, a linear relation exists between T and $b(T)$ with a slope k.

XX dynamic model

Geeraerd's model

The inactivation model described by Geeraerd et al. (2000) is based on the assumption that the inactivation process follows a first order kinetics with rate k.

$$\frac {dN}{dt} = - k \cdot N$$

However, the parameter k with two parameters to include for shoulder (α) and tail (γ) effects.

$$k = k_{max} \cdot \alpha \cdot \gamma$$

where k_{max} stands for the maximum inactivation rate of the microorganism for some conditions given. The shoulder parameter is defined as

$$\alpha = \frac {1}{1+C}$$

where C is a dummy components that limits the microbial inactivation. It is assumed that this component also follows a first order kinetics with rate k_{max}:

$$\frac {dC}{dt} = - k_{max} \cdot C$$

The tail parameter is defined by the equation

$$ \gamma = 1 - \frac{N_{res}}{N} $$

where N_{res} stands for the residual number of microorganism after the treatment.

Prediction of microbial inactivation

The **inactivation** package can predict the inactivation process of a microorganism through the `predict_inactivation()` function, which solves the differential equation required using the `lsoda` integrator.

The `predict_inactivation()` function requires four input variables:

- `simulation_model`
- `times`
- `parms`
- `temp_profile`

`simulation_model` is a string identifying the model to use. A list of the availables models can be obtained by calling the function `get_model_data()` without any argument.

```
get_model_data()
```

```
## [1] "Bigelow"             "Bigelow_linearized"  "Weibull_Mafart"
## [4] "Weibull_Mafart_full" "Geeraerd"            "Weibull_Peleg"
## [7] "Weibull_Peleg_full"
```

In this example Geeraerd's model will be used, so the value `Geeraerd` will be assigned.

```
example_model <- "Geeraerd"
```

`times` is a numeric vector which specifies at which values of time the results will be output. It does not define the points at which the numerical integration is done, so changing it values will not affect the values calculated at individual time points.

```
times <- seq(0, 5, length=100)
```

`parms` is a named numeric vector specifying the values of the model parameters, as well as initial values for the integration. A call to the function `get_model_data()` with input variable a model identifier provides several data concerning this model as a list. Namely, its parameters with the header *parameters* and its variables with the header *variable*.

```
model_data <- get_model_data(example_model)
print(model_data$parameters)
```

```
## [1] "D_R"      "z"        "N_min"    "temp_ref"
```

```
print(model_data$variable)
```

```
## [1] "N"   "C_c"
```

`parms` needs to define values for all the parameters with names as provided by the function

`get_model_data()`. Initial values for the variables must be named as the number of the variable plus the string “0”.

```
model_parms <- c(D_R = 1,
                 z = 10,
                 N_min = 100,
                 temp_ref = 100,
                 N0 = 100000,
                 C_c0 = 1000
                 )
```

The last input variable required for the function is a `data.frame` defining a discrete temperature profile. Values of the temperature at time points not provided will be approximated by using linear interpolation. A trapezoidal temperature profile will be used in this example.

```
temperature_profile <- data.frame(time = c(0, 1.25, 2.25, 4.6),
                                  temperature = c(70, 105, 105, 70))
```

The prediction is calculated by calling the function `predict_inactivation()` with the aforementioned parameters.

```
prediction_results <- predict_inactivation(example_model, times, model_parms, temperature_profile)
```

The value returned by the function is a list of class `SimulInactivation`. This list has four entries which provide enough information to reproduce the calculation:

- model: a string with the identifier of the model used for the simulation.
- model_parameters: a named numeric list with the model parameters used.
- temp_approximations: a list with two entries, each providing the functions used to approximate the temperature at each time point.
- simulation: a data frame with the results of the simulation

The entry *simulation* provides a data frame with the results calculated.

```
head(prediction_results$simulation)
```

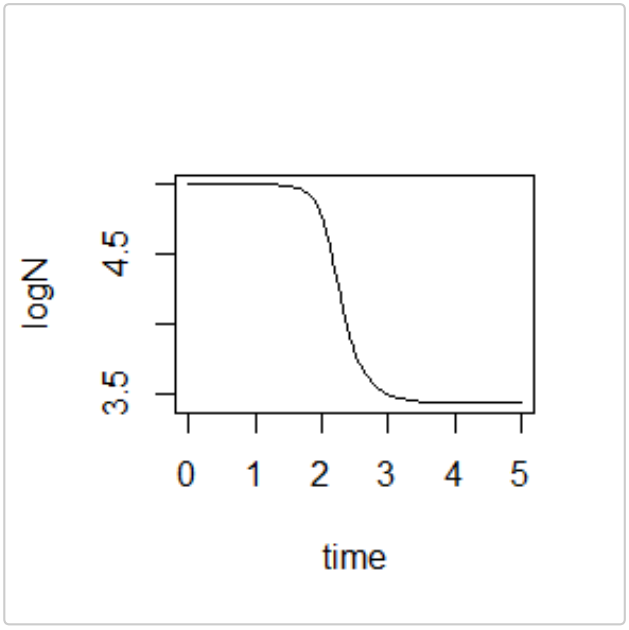
```
##         time         N       C_c     logN         S          logS
## 1 0.00000000 100000.00 1000.0000 5.000000 1.0000000  0.000000e+00
## 2 0.05050505  99999.99  999.8620 5.000000 0.9999999 -5.984094e-08
## 3 0.10101010  99999.97  999.6707 5.000000 0.9999997 -1.427680e-07
## 4 0.15151515  99999.94  999.4074 5.000000 0.9999994 -2.570217e-07
## 5 0.20202020  99999.90  999.0421 5.000000 0.9999990 -4.155888e-07
## 6 0.25252525  99999.85  998.5364 4.999999 0.9999985 -6.352974e-07
```

On the first column, the times at which results are output are given. They are equal to the values provided by the `times` input variable.

The following columns provide the results obtained for the variables of the models. In this case, `N` and `C_c`. The last three columns provide the values of \(\mathrm{log}_{10}(N)\), \(S\) and \(\mathrm{log}_{10}(S)\).

The `SimulInactivation` object includes S3 methods for the plotting of its results. Calling the function `plot()` with the object as arguments produces this plot.

```
plot(prediction_results)
```



Fitting of isothermal data

The package **inactivation** can fit the parameters of an inactivation model to a set of isothermal experimental data through the function `fit_isothermal_inactivation()`. This function makes use of the `nls` function within the `stats` package to fit the parameters of an inactivation model to a set of experimental data.

The experimental data to fit is provided by the `data.frame` variable `death_data`. It has to contain at least three columns providing the time at which the observation was made (*time*), the temperature of the experimetn (*temp*) and the logarithmic difference (*log_diff*) observed with respect to the orginal population. The example data set `isothermal_example` will be used for this demonstration.

```
data(isothermal_example)
```

The model to use for the adjustment is defined by the character variable `model_name`. A list of the valid model keys for isothermal adjustment can be obtained by calling the function `get_isothermal_model_data()` without any arguments.

```
get_isothermal_model_data()
```

```
## [1] "Weibull_Mafart" "Weibull_Pelec"  "Bigelow"
```

Bigelow's model, whose key is `"Bigelow"`, will be used in this example.

```
model_name <- "Bigelow"
```

The names of the parameters of an inactivation model can be obtained by calling the function `get_isothermal_model_data()` with the model key as argument.

```
model_data <- get_isothermal_model_data(model_name)
model_data$params
```

```
## [1] "z"        "D_ref"    "ref_temp"
```

Therefore, the isothermal Bigelow model requires the definition of three parameters: `z`, `D_ref` and

`ref_temp`. The `fit_isothermal_inactivation()` function allows to distinguish between model parameters known and unknown. The model parameters known are set by the input argument `known_parameters`, which has to be a `list`. The reference temperature is usually set, so we will fix it to 100ºC in this example.

```
known_params = list(ref_temp = 100)
```

Due to the use of the `nls` functions, initial points are required for the unknown model parameters. These initial values are provided by the input argument `starting_point`. This variable needs to be a named numeric vector.

```
starting_point <- c(z = 10,
                    D_ref = 1
                    )
```

The `fit_isothermal_inactivation()` functions allows to make the adjustment based on the minimization of the error of either the logarithmic count or the total number of microorganisms. In this example, the adjustment will be based on the minimization of the error of the logarithmic count, so the `adjust_log` argument will be set to `TRUE`.

```
adjust_log <- TRUE
```

The adjustment is made by calling the `fit_isothermal_inactivation()` function with the arguments which have just been defined. This function returns a list of class `IsoFitInactivation`.

```
iso_fit <- fit_isothermal_inactivation(model_name, isothermal_example,
                                       starting_point, adjust_log, known_params)
```

The returned object has four entries, which provide information about the adjustment process as well enough information to reproduce the calculation:

- The information of the adjustment process is provided under the entry `nls`, where the object returned by the `nls` function is saved.

```
iso_fit$nls
```

```
## Nonlinear regression model
##   model: log_diff ~ Bigelow_iso(time, temp, z, D_ref, ref_temp)
##    data: death_data
##     z D_ref
## 11.00  1.25
##  residual sum-of-squares: 13.89
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 2.437e-09
```

- The values of the parameters fixed under the input argument `known_arguments` as well as those calculated by the adjustment are provided as a list. This list can be accessed under the entry `parameters`.

```
iso_fit$parameters
```

```
## $ref_temp
```

```
## [1] 100
##
## $z
## [1] 11.00493
##
## $D_ref
## [1] 1.249546
```

- The key of the model used for the adjustment is provided under the header `model`.

```
iso_fit$model
```

```
## [1] "Bigelow"
```

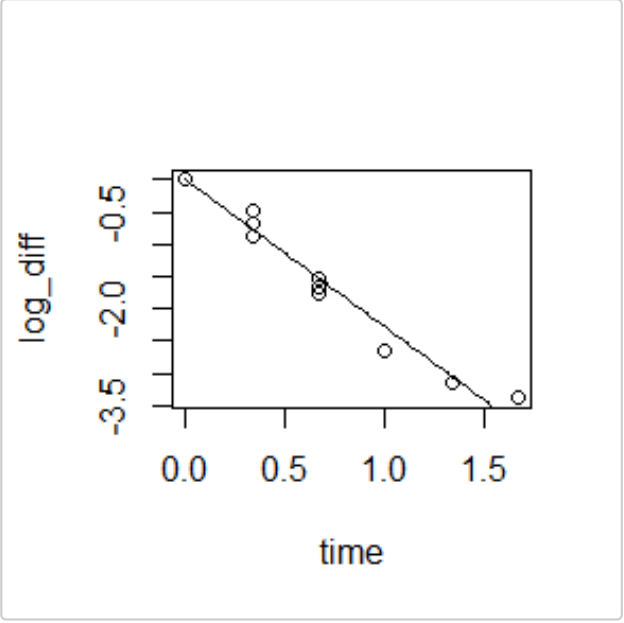
- The data used for the adjustment is saved under the header `data`.

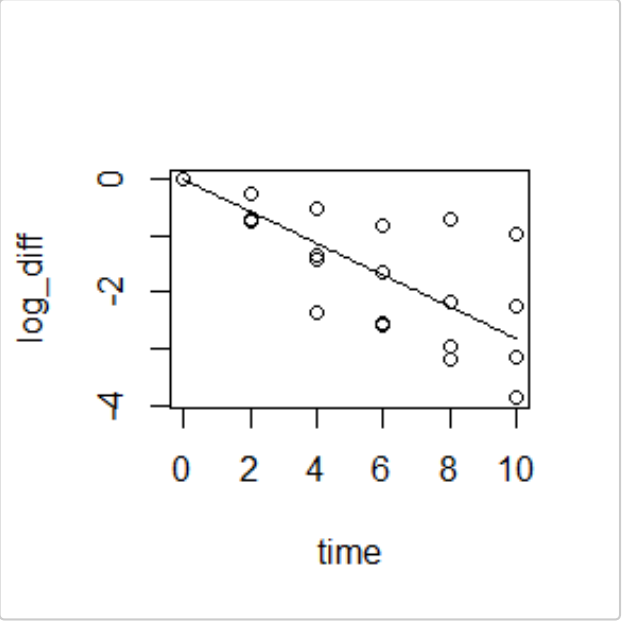
```
head(iso_fit$data)
```

```
##        time temp   log_diff
## 1 0.0000000  105  0.0000000
## 2 0.3333333  105 -0.6777807
## 3 0.6666667  105 -1.6777807
## 4 0.0000000  105  0.0000000
## 5 0.3333333  105 -0.8822398
## 6 0.6666667  105 -1.5300573
```

The class `IsoFitInactivation` implements S3 methods for plot, which allow the creation of a comparison plot between the best prediction and the experiment results.

```
plot(iso_fit)
```





Fitting of dynamic experiments

The **inactivation** package implements the function `fit_dynamic_inactivation()`, which allows the adjustment of inactivation models to experimental data with time dependent temperature profiles. This function requires several input arguments:

- The experimental data to fit is provided by the variable `experiment_data`. It has to be a `data.frame` with at least a column named *time*, which provides the time at which the measurement was taken, and another one named *N*, with the number of microorganisms counted at that time.
- The model to use for the fitting is defined by the character variable `simulation_model`. A list of the availables models for dynamic adjustment can be obtained by calling the function `get_model_data()` without any input argument.
- The temperature profile of the experiment is given by the `data.frame` `temp_profile`. This variable provides the values of the temperature at discrite time points. Values between the values of time provided are calculated by linear interpolation.
- The values of the model parameters that are known and, thus, do not need to be adjusted are given by the input argument `fixed_parameters`. This variable is a named numeric vector whose names must match the predefined ones for the model parameters. These predefined names can be obtained by calling the function `get_model_data()` with the model key as argument.
- The model parameters that need to be adjusted are calculated by nonlinear optimization. Therefore, initial points and upper and lower bounds need to be defined for each one of them. This is made through the variables `starting_points`, `upper_bounds` and `lower_bounds`. These variables share format with the input argument `fixed_parameters`. In case of a boundless adjustment of any of the variables, `Inf` must be assigned to the desired entries of `upper_bounds` or `lower_bounds`.
- As well as the model parameters, the initial number of microorganism must be defined as either a fixed parameter or a parameter to fit. This definition is analogous to the one of the model parameters, although its name must be equal to the one provided by the function `get_model_data()` plus a caracter `"0"` . For example, the initial value of the variable `"N"` must be named `"N0"`.
- The adjustment can be based on the minimization of the error of the total count of microorganism (\(N\)) or the logarithmic count (\(\mathrm{log}_{10}N\)). This distinction is made by the boolean input argument `minimize_log`. In case it is `TRUE`, the error of the logarithmic count is minimized.

The data set `inactivation_example` included in the **inactivation** package will be used for the demonstration of the capabilities of the `fit_dynamic_inactivation()` function.

```
data(inactivation_example)
```

Calling the function `get_model_data()` without any input argument allows as to identify the key corresponding to this model.

```
get_model_data()
```

```
## [1] "Bigelow"             "Bigelow_linearized"  "Weibull_Mafart"
## [4] "Weibull_Mafart_full" "Geeraerd"            "Weibull_Peleg"
## [7] "Weibull_Peleg_full"
```

The Weibull-Peleg model, whose key is *Weibull_Peleg* will be used in this example.Therefore, we assign it to variable `simulation_model`.

```
simulation_model <- "Weibull_Peleg"
```

A dummy trapezoidal temperature profile will be used for the model fitting.

```
dummy_temp <- data.frame(time = c(0, 1.25, 2.25, 4.6),
                         temperature = c(70, 105, 105, 70))
```

In order to know the names of the model parameters, the function `get_model_data()` is called with the model key as argument.

```
model_data <- get_model_data("Weibull_Peleg")
model_data$parameters
```

```
## [1] "k_b"       "temp_crit" "n"
```

Therefore, this model requires three parameters. The parameter *temp_crit* will be considered known equal to 100ºC in this example.

```
fixed_parameters = c(temp_crit = 100)
```

The rest of the parameters, as well as the initial number of microorganism, will be considered unknown. Initial values relatively close to the expected solution will be used.

```
starting_points <- c(n = 1,
                     k_b = 0.25,
                     N0 = 1e+05)
```

Reasonable lower and upper bounds will be defined, so variables cannot be negative or unrealistically large.

```
upper_bounds <- c(n = 2,
                  k_b = 1,
                  N0 = Inf)

lower_bounds <- c(n = 0,
                  k_b = 0,
                  N0 = 1e4)
```

Care must be taken when setting initial points and bounds, because if they are unreasonable the fitting algorithm will fail.

For this example, the error in the total number of microorganism will be optimized. We, thus, define the variable `minimize_log` accordingly:

```
minimize_log = TRUE
```

The adjustment is made by calling the function `fit_dynamic_inactivation()` with the variables which have been just defined.

```
dynamic_fit <- fit_dynamic_inactivation(inactivation_example, simulation_model, dummy_temp,
                                        starting_points, upper_bounds, lower_bounds,
                                        fixed_parameters, minimize_log)
```

The function returs a list of class `FitInactivation` with three entries:

- *fit_results* provides the instance of `modFit` returned by the `FME::modFit()` function.
- under *best_prediction* an instance `SimulInactivation` with the results of the best prediction is saved.
- the header *data* contains the experimental data used for the fit.

The former provides the instance of `modFit` returned by the `FME::modFit()` function. The latest provides an instance of `"SimulInactivation"` with the results of the best prediction.

```
names(dynamic_fit)
```

```
## [1] "fit_results"     "best_prediction" "data"
```

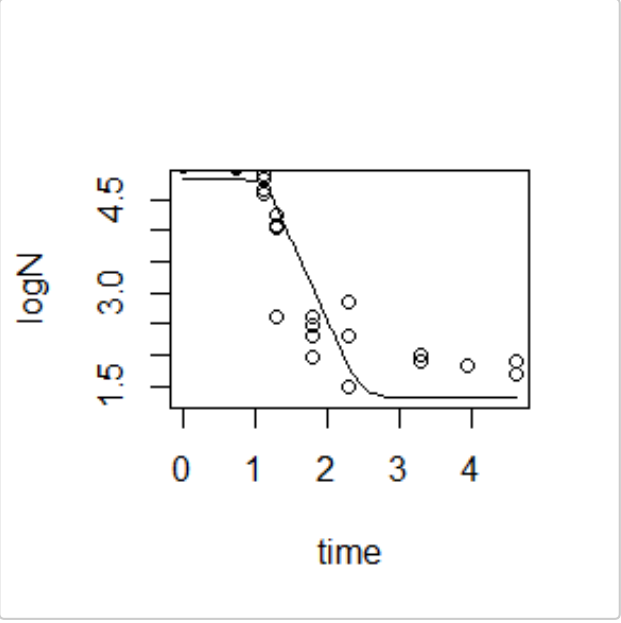
The parameters of the model calculated are saved under the entry`"model_parameters"` of the `"SimulInactivation"` instance.

```
dynamic_fit$best_prediction$model_parameters
```

```
## $n
## [1] 1.00067
##
## $k_b
## [1] 0.4948966
##
## $N0
## [1] 66539.8
##
## $temp_crit
## [1] 100
```

The `FitInactivation` object includes S3 methods for the creation of a comparison plot between the experimental results and the prediction made by the model parameters adjusted.

```
plot(dynamic_fit)
```



References

XX Ordenar referencias
XX Homogeneizar formatos referencias

Karline Soetaert, Thomas Petzoldt, R. Woodrow Setzer (2010). *Solving Differential Equations in R: Package deSolve Journal of Statistical Software*, 33(9), 1–25. URL http://www.jstatsoft.org/v33/i09/.

Soetaert, Karline and Petzoldt, Thomas, 2010. *Inverse Modelling, Sensitivity and Monte Carlo Analysis in R Using Package FME. Journal of Statistical Software*, 33(3), 1-28. URL http://www.jstatsoft.org/v33/i03/.

Mafart, P., Couvert, O., Gaillard, S., & Leguerinel, I. (2002). On calculating sterility in thermal preservation methods: Application of the Weibull frequency distribution model. International Journal of Food Microbiology, 72, 107–113.

Geeraerd A.H., C.H. Herremans and J.F. Van Impe 2000. “Structural model requirements to describe microbial inactivation during a mild heat treatment.” International Journal of Food Microbiology, 59, 185-209.