Package: OenoKPM (via r-universe)

September 6, 2024

Title Modeling the Kinetics of Carbon Dioxide Production in Alcoholic Fermentation

Version 2.4.1

Description Developed to help researchers who need to model the kinetics of carbon dioxide (CO2) production in alcoholic fermentation of wines, beers and other fermented products. The following models are available for modeling the carbon dioxide production curve as a function of time: 5PL, Gompertz and 4PL. This package has different functions, which applied can: perform the modeling of the data obtained in the fermentation and return the coefficients, analyze the model fit and return different statistical metrics, and calculate the kinetic parameters: Maximum production of carbon dioxide; Maximum rate of production of carbon dioxide; Moment in which maximum fermentation rate occurs; Duration of the latency phase for carbon dioxide production; Carbon dioxide produced until maximum fermentation rate occurs. In addition, a function that generates graphs with the observed and predicted data from the models, isolated and combined, is available. Gava, A., Borsato, D., & Ficagna, E. (2020).` ` Effect of mixture of fining agents on the fermentation kinetics of base wine for sparkling wine production: Use of methodology for modeling". <doi:10.1016/j.lwt.2020.109660>.

Depends R (>= 4.0.0)

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Imports ggplot2,minpack.lm,openxlsx,grDevices,ggpubr, grid, gridExtra

NeedsCompilation no

Author Angelo Gava [aut, cre] (<https://orcid.org/0000-0003-0338-6511>)

Maintainer Angelo Gava <gava.angelogava@gmail.com>

Date/Publication 2024-04-08 19:20:10 UTC

Repository https://angelogava.r-universe.dev

RemoteUrl https://github.com/cran/OenoKPM

RemoteRef HEAD

RemoteSha 770892cd5ebdfdd2c6c682ac0ab6b613e6214d0e

Contents

---

kp *Calculates kinetic parameters as a function of model fit for $CO2$ production as a function of time*

Description

A function that, based on the observed data, the independent variable (e.g. time in h) and the dependent variable (e.g. CO_2 production in g L^{-1}), performs the modeling of the fermentation curve based on the chosen model (**5PL**, **Gompertz**, or **4PL**).

Next, the coefficients are used in mathematical formulas to obtain the following kinetic parameters:

$\mathbf{t}_{Lag}$ - Duration of the latency phase for CO_2 production;

$\mathbf{V}_{max}$ - Maximum rate of production of CO_2;

$\mathbf{t}_{V_{max}}$ - Moment in which maximum fermentation rate occurs;

$\mathbf{CO2}_{V_{max}}$ - CO_2 Produced until Maximum fermentation rate occurs;

$\mathbf{Y}_{max}$ - Maximum production of carbon dioxide (CO_2);

Usage

```
kp(
  data,
  model,
  save.xls = FALSE,
  dir.save,
  xls.name,
  startA,
  startB,
  startC,
  startD,
  startG
)
```

Arguments

`data`	Data frame to be analyzed. The data frame must be in the following order: • **First**: All columns containing the independent variable (e.g. *time in hours*) • **Second**: All columns containing dependent variables (e.g. CO_2 *g* L^{-1} *production*) • **Header**: Columns must contain a header. If the treatment **ID** is in the header, this **ID** will be used to **identify** the coefficients and kinetic parameters for each analyzed curve.
`model`	Model to be adjusted. Argument for model: • **Model = 1**. 5PL Model (five-parameter logistic (5PL) model). • **Model = 2**. Gompertz Model. • **Model = 3**. 4PL Model (four-parameter logistic (4PL) model).
`save.xls`	If TRUE, an xlsx file containing the coefficients and kinetic parameters will be saved in the working directory. If FALSE, the xlsx file will not be saved.
`dir.save`	Directory path where the xlsx file is to be saved.
`xls.name`	File name. Must contain the format. For example, "Parameters.xlsx".
`startA`	Starting estimate of the value of A for model.
`startB`	Starting estimate of the value of B for model.
`startC`	Starting estimate of the value of C for model.
`startD`	Starting estimate of the value of D for model.
`startG`	Starting estimate of the value of G for model.

Details

Curve fitting from the observed data is performed by the nlsLM() function in the 'minpack.lm' package.

You can see our article for more details on the mathematical formulas used to obtain each kinetic parameter (Gava *et al*., 2020). In addition, feel free to use it as a reference in your works.

Value

The analyzed model **coefficients** and the calculated **kinetic parameters** are returned in a data.frame. In addition, a **"Parameters.xlsx" file** can be generated, containing the coefficients and kinetic parameters of each studied fermentation curve.

Author(s)

Angelo Gava

References

Gava, A., Borsato, D., & Ficagna, E. (2020). Effect of mixture of fining agents on the fermentation kinetics of base wine for sparkling wine production: Use of methodology for modeling. *LWT*, *131*, 109660. doi:10.1016/j.lwt.2020.109660

Zwietering, M. H., Jongenburger, I., Rombouts, F. M., & Van't Riet, K. J. A. E. M. (1990). Modeling of the bacterial growth curve. *Applied and environmental microbiology*, *56*(6), 1875-1881. doi:10.1128/aem.56.6.18751881.1990

Examples

```
#Creating a data.frame.
#First, columns containing independent variable.
#Second, columns containing dependent variable.
#The data frame created presents two
#fermentation curves for two yeasts with
#different times and carbon dioxide production.

df <- data.frame('Time_Yeast_A' = seq(0,280, by=6.23),
                 'Time_Yeast_B' = seq(0,170, by=3.7777778),
                 'CO2_Production_Yeast_A' = c(0,0.97,4.04,9.62,13.44,17.50,
                                              24.03,27.46,33.75,36.40,40.80,
                                             44.24,48.01,50.85,54.85,57.51,
                                             61.73,65.43,66.50,72.41,75.47,
                                             77.22,78.49,79.26,80.31,81.04,
                                             81.89,82.28,82.56,83.13,83.62,
                                             84.11,84.47,85.02,85.31,85.61,
                                             86.05,86.27,85.29,86.81,86.94,
                                             87.13,87.33,87.45,87.85),
                 'CO2_Production_Yeast_B' = c(0,0.41,0.70,3.05,15.61,18.41,
                                              21.37,23.23,28.28,41.28,43.98,
                                              49.54,54.43,60.40,63.75,69.29,
                                              76.54,78.38,80.91,83.72,84.66,
                                              85.39,85.81,86.92,87.38,87.61,
                                              88.38,88.57,88.72,88.82,89.22,
                                              89.32,89.52,89.71,89.92,90.11,
                                              90.31,90.50,90.70,90.90,91.09,
                                              91.29,91.49,91.68,91.88))

#Using the kp() function to find the
#coefficients and kinetic parameters
#according to the adopted model.

kp(data = df,
model = 1,
startA = 0,
startB = 1.5,
startC = 500,
startD = 92,
startG = 1500,
save.xls = FALSE) #5PL Model adopted

kp(data = df,model = 2,
startA = 92,
startB = 1.5,
startC = 0,
startD = NA,
startG = NA,
```

```
save.xls = FALSE) #Gompertz Model adopted

kp(data = df,
startA = 0,
startB = 2.5,
startC = 10,
startD = 92,
startG = NA,
model = 3,
save.xls = FALSE) #4PL Model adopted

#Saving an xlsx file. In this example,
#we will use saving a temporary file in
#the temporary file directories.
```

`metrics` *Performs the modeling of the observed data and returns the fit metrics of the studied model*

Description

A function that, based on the observed data, the independent variable (e.g. time in h) and the dependent variable (e.g. CO_2 production in g L^{-1}), performs the modeling of the fermentation curve based on the chosen model (**5PL**, **Gompertz**, or **4PL**) and returns the model fit **metrics**.

As a result, the fit metrics for the chosen model are returned in the form of data.frame: **Correlation**, $\mathbf{R}^2$,**Residual sum of squares (**$\mathbf{RSS}_{min}$**)** and **Residual standard error**.

Usage

```
metrics(
  data,
  model,
  save.xls = FALSE,
  dir.save,
  xls.name,
  startA,
  startB,
  startC,
  startD,
  startG
)
```

Arguments

`data` Data frame to be analyzed. The data frame must be in the following order:

- **First**: All columns containing the independent variable (e.g. *time in hours*)

- **Second**: All columns containing dependent variables (e.g. $CO_2\ g\ L^{-1}$ *production*)
- **Header**: Columns must contain a header. If the treatment **ID** is in the header, this **ID** will be used to **identify** the metrics for each analyzed curve.

`model` Model to be adjusted. Argument for model:

- **Model = 1**. 5PL Model (five-parameter logistic (5PL) model)
- **Model = 2**. Gompertz Model
- **Model = 3**. 4PL Model (four-parameter logistic (4PL) model)

`save.xls` If TRUE, an xlsx file containing the metrics will be saved in the working directory. If FALSE, the xlsx file will not be saved.

`dir.save` Directory path where the xlsx file is to be saved.

`xls.name` File name. Must contain the format. For example, "Metrics.xlsx".

`startA` Starting estimate of the value of A for model.

`startB` Starting estimate of the value of B for model.

`startC` Starting estimate of the value of C for model.

`startD` Starting estimate of the value of D for model.

`startG` Starting estimate of the value of G for model.

Details

Curve fitting from the observed data is performed by the nlsLM() function in the 'minpack.lm' package.

Value

The **metrics** from the analyzed model are returned in a **data.frame**. In addition, a **"Metrics.xlsx" file** can be generated, containing the **model fit metrics** for each fermentation curve studied: **Correlation**; $\mathbf{R}^2$; **Residual standard error**; **Residual sum of squares** ($\mathbf{RSS}_{min}$).

Author(s)

Angelo Gava

Examples

```
#Creating a data.frame.
#First, columns containing independent variable.
#Second, columns containing dependent variable.
#The data frame created presents two
#fermentation curves for two yeasts with
#different times and carbon dioxide production.

df <- data.frame('Time_Yeast_A' = seq(0,280, by=6.23),
                 'Time_Yeast_B' = seq(0,170, by=3.7777778),
                 'CO2_Production_Yeast_A' = c(0,0.97,4.04,9.62,13.44,17.50,
                                              24.03,27.46,33.75,36.40,40.80,
                                             44.24,48.01,50.85,54.85,57.51,
```

```
                                          61.73,65.43,66.50,72.41,75.47,
                                          77.22,78.49,79.26,80.31,81.04,
                                          81.89,82.28,82.56,83.13,83.62,
                                          84.11,84.47,85.02,85.31,85.61,
                                          86.05,86.27,85.29,86.81,86.94,
                                          87.13,87.33,87.45,87.85),
                  'CO2_Production_Yeast_B' = c(0,0.41,0.70,3.05,15.61,18.41,
                                           21.37,23.23,28.28,41.28,43.98,
                                           49.54,54.43,60.40,63.75,69.29,
                                           76.54,78.38,80.91,83.72,84.66,
                                           85.39,85.81,86.92,87.38,87.61,
                                           88.38,88.57,88.72,88.82,89.22,
                                           89.32,89.52,89.71,89.92,90.11,
                                           90.31,90.50,90.70,90.90,91.09,
                                           91.29,91.49,91.68,91.88))

#Using the metrics() function to find the
#model fit metrics

metrics(data = df,
model = 1,
startA = 0,
startB = 1.5,
startC = 500,
startD = 92,
startG = 1500,
save.xls = FALSE) #5PL Model adopted

metrics(data = df,
model = 2,
startA = 92,
startB = 1.5,
startC = 0,
startD = NA,
startG = NA,
save.xls = FALSE) #Gompertz Model adopted

metrics(data = df,
model = 3,
startA = 0,
startB = 2.5,
startC = 10,
startD = 92,
startG = NA,
save.xls = FALSE) #4PL Model adopted

#Saving an xlsx file. In this example,
#we will use saving a temporary file in
#the temporary file directories.
```

plot_fit *Plot graphs with observed data and predicted data from models*

Description

A function that, based on the observed data, the independent variable (e.g. time in h) and the dependent variable (e.g. CO_2 production in g L^{-1}), performs the modeling of the fermentation curve based on the chosen model(s) (**5PL**, **Gompertz**, or/and **4PL**).

From the observed data and predicted data, whether from one or all of the available models, this function will plot a graph for each fermentation curve evaluated. The chart will have the following basic structure:

X axis: *fermentation time*

Y axis: *CO_2 production*

Observed data: *Scatterplot with dots*. Plot with geom_point function from ggplot2 package.

Predicted data: *Smoothed line*. Plot with the stat_smooth function from the ggplot2 package.

Usage

```
plot_fit(
  data,
  models,
  startA,
  startB,
  startC,
  startD,
  startG,
  col = "black",
  col1 = "red",
  col2 = "cornflowerblue",
  col3 = "forestgreen",
  axisX = "Time (hours)",
  axisY = expression(paste("CO"["2"] * " Production (g L"^{
     "-1"
 } * ")")),
  breaksX = waiver(),
  limitsX = NULL,
  breaksY = waiver(),
  limitsY = NULL,
  font = "serif",
  font.size = 14,
  legend.position = "top",
  show.R2 = FALSE,
  save.PDF = FALSE,
  dir.save,
  dir.name = "Graphics",
```

```
  width.PDF = 15,
  height.PDF = 12,
  width.PDF2 = 25,
  height.PDF2 = 18
)
```

Arguments

data	Data frame to be analyzed. The data frame must be in the following order: • **First**: All columns containing the independent variable (e.g. *time in hours*) • **Second**: All columns containing dependent variables (e.g. CO_2 *g* L^{-1} *production*) • **Header**: Columns must contain a header. If the treatment **ID** is in the header, this **ID** will be used to name the graphics PDF files for each analyzed curve.
models	Model or models to be adjusted: • **Models = 1**. Only the 5PL Model. • **Models = 2**. Only the Gompertz Model. • **Models = 3**. Only the 4PL Model. • **Models = 4**. 5PL and Gompertz Models. • **Models = 5**. 5PL and 4PL Models. • **Models = 6**. Gompertz and 4PL Models. • **Models = 7**. 5PL, Gompertz and 4PL Models.
startA	Starting estimate of the value of A for model.
startB	Starting estimate of the value of B for model.
startC	Starting estimate of the value of C for model.
startD	Starting estimate of the value of D for model.
startG	Starting estimate of the value of G for model.
col	Plot color of observed data in points. For example, "black".
col1	Plot color of predicted data from **model 1** (*5PL Model*). For example, "red".
col2	Plot color of predicted data from **model 2** (*Gompertz Model*). For example, "blue".
col3	Plot color of predicted data from **model 3** (*4PL Model*). For example, "green".
axisX	X Axis Title. Character vector (or expression).
axisY	Y Axis Title. Character vector (or expression).
breaksX	One of: • Use ggplot2::waiver() for the default X-axis breaks calculated by the transform object. • Numerical vector of X-Axis scale positions. For example, seq(0,200,20).
limitsX	One of: • NULL to use the default X-Axis scaling range.

- A numeric vector of length two, giving the limits of the X-axis scale. For example, c(0,200).

`breaksY` One of:

- Use ggplot2::waiver() for the default Y-axis breaks calculated by the transform object.
- A numerical vector of Y-Axis scale positions. For example, seq(0,100,10).

`limitsY` One of:

- NULL to use the default Y-Axis scaling range.
- A numeric vector of length two, giving the limits of the Y-axis scale. For example, c(0,100).

`font` Base font family

`font.size` Base font size, given in pts.

`legend.position`
The position of the caption ("none", "left", "right", "bottom", "top", or a numeric vector of two elements (X,Y).

`show.R2` If TRUE, plots the R^2 of the plotted predicted models on the graph. If FALSE, do not plot the R^2 of the plotted predicted models.

`save.PDF` If TRUE, create a folder (directory) and save each graphic in PDF format. If FALSE, it does not create a directory or save the graphics in PDF.

`dir.save` Path of the directory in which a new folder (directory) will be created for saving graphics in PDF format.

`dir.name` Folder name (directory name) to be created within the working directory for saving PDF graphics. Character vector.

`width.PDF` Width, in cm, of the graphic to be saved in a PDF file.

`height.PDF` Height, in cm, of the graphic to be saved in a PDF

`width.PDF2` Width, in cm, of the multiplot graphic to be saved in a PDF file.

`height.PDF2` Height, in cm, of the multiplot graphic to be saved in a PDF

Details

Curve fitting from the observed data is performed by the nlsLM() function in the 'minpack.lm' package.

Graphs are plotted using the various functions in the 'ggplot2' package.

Value

Elegant graphics plotted according to observed and predicted data. In addition, a folder (directory) can be created, in which the PDF graphics will be saved, if desired. In this folder, the graph of each analyzed fermentation curve is saved in PDF format, with the dimensions stipulated in the *width.PDF* and *height.PDF* arguments. The name of each PDF file will be extracted from the header of the dependent variable used for the graph. See more in the examples.

Author(s)

Angelo Gava

Examples

```
#################Example 1#################
#Using only required arguments

#Creating a data.frame.
#First, columns containing independent variable.
#Second, columns containing dependent variable.
#The data frame created presents two
#fermentation curves for two yeasts with
#different times and carbon dioxide production.

df <- data.frame('Time_Yeast_A' = seq(0,280, by=6.23),
 'Time_Yeast_B' = seq(0,170, by=3.7777778),
 'CO2_Production_Yeast_A' = c(0,0.97,4.04,9.62,13.44,17.50,
                              24.03,27.46,33.75,36.40,40.80,
                              44.24,48.01,50.85,54.85,57.51,
                              61.73,65.43,66.50,72.41,75.47,
                              77.22,78.49,79.26,80.31,81.04,
                              81.89,82.28,82.56,83.13,83.62,
                              84.11,84.47,85.02,85.31,85.61,
                              86.05,86.27,85.29,86.81,86.94,
                              87.13,87.33,87.45,87.85),
 'CO2_Production_Yeast_B' = c(0,0.41,0.70,3.05,15.61,18.41,
                              21.37,23.23,28.28,41.28,43.98,
                              49.54,54.43,60.40,63.75,69.29,
                              76.54,78.38,80.91,83.72,84.66,
                              85.39,85.81,86.92,87.38,87.61,
                              88.38,88.57,88.72,88.82,89.22,
                              89.32,89.52,89.71,89.92,90.11,
                              90.31,90.50,90.70,90.90,91.09,
                              91.29,91.49,91.68,91.88))

#Using the plot_fit function to
#generate elegants graphs PDF files
#containing both observed data and
#predicted data.


#Graph plotted only with Model 5PL
#fit (models = 1)

plot_fit(data = df,
         models = 1,
         startA = 0,
         startB = 1.5,
         startC = 500,
         startD = 92,
         startG = 1500)


#Graph plotted with 5PL and Gompertz
#model fits (models = 4)
```

```
plot_fit(data = df,
         models = 4,
         startA = 0,
         startB = 1.5,
         startC = 500,
         startD = 92,
         startG = 1500)



#################Example 2#################
#Using the various function arguments to
#customize the graph.

#Creating a data.frame.
#First, columns containing independent variable.
#Second, columns containing dependent variable.
#The data frame created presents two
#fermentation curves for two yeasts with
#different times and carbon dioxide production.

df <- data.frame('Time_Treatment_A' = seq(0,200, by=6.45),
                 'Time_Treatment_B' = seq(0,200, by=6.45),
                 'CO2_Production_Treatment_A' = c(0,0.47,0.78,3.23,19.15,22.86,
                                                  26.81,29.36,36.14,52.61,55.58,
                                                  61.38,66.25,71.83,74.8,78.88,
                                                  83.47,84.48,85.94,87.45,87.98,
                                                  88.42,88.68,89.40,89.72,89.87,
                                                  90.41,90.51,90.62,90.70,91.05,
                                                  91.185),
                 'CO2_Production_Treatment_B' = c(0,0.19,0.39,1.36,9.23,11.29,
                                                  13.58,15.06,19.34,30.92,33.28,
                                                  37.98,42.14,47.17,50.00,54.28,
                                                  60.92,62.80,65.54,69.74,71.52,
                                                  73.07,73.98,76.75,77.79,78.70,
                                                  80.65,81.48,82.07,82.47,84.04,
                                                  84.60))

#Using the plot_fit function to
#generate elegants graphs PDF files
#containing both observed data and
#predicted data.

#Graph plotted only with Model 5PL
#fit (models = 1)
#Do not show R^2

plot_fit(data = df,
         startA = 0,
         startB = 1.5,
         startC = 500,
         startD = 92,
```

```
          startG = 1500,
          models = 1,
          col = "red", #Color of observed data (points)
         col1 = "blue", #Predicted data color from model 1 (line). Model = 1 <- 5PL Model
          axisX =  "Fermentation time (h)", #Title X-Axis
          axisY = "Carbon dioxide production (g/L)", #Title Y-Axis
          breaksX = seq(0,200,20), #X-Axis scale (positions). 0,20,40,60,80,...
          limitsX = c(0,200), #X-Axis Limits
          breaksY = seq(0,90,5),#Y-Axis scale (positions). 0,5,10,15,20,...
          limitsY = c(0,95), #Y-Axis Limits
          font = "serif",
          font.size = 12,
          legend.position = "right",
          show.R2 = FALSE) #Do not show R^2



#Graph plotted with 5PL and 4PL
#model fits (models = 5)
#Show R^2
## Not run:
plot_fit(data = df,
         models = 5,
         startA = 0,
         startB = 1.5,
         startC = 500,
         startD = 92,
         startG = 1500,
         col = "#000000", #Color of observed data (points)
        col1 = "#FF0000", #Predicted data color from model 1 (line). Model = 1 <- 5PL Model
        col3 = "#0B6121",#Predicted data color from model 3 (line). Model = 3 <- 4PL Model
         axisX =  "Time (h)", #Title X-Axis
         axisY = "CO2 production (g/L)", #Title Y-Axis
         breaksX = seq(0,200,20), #X-Axis scale (positions). 0,20,40,60,80,...
         limitsX = c(0,200), #X-Axis Limits
         breaksY = seq(0,90,10),#Y-Axis scale (positions). 0,10,20,30,40,...
         limitsY = c(0,95), #Y-Axis Limits
         font = "serif",
         font.size = 14,
         legend.position = "bottom",
         show.R2 = TRUE) #Show R^2

## End(Not run)

#Graph plotted with 5PL, Gompertz and 4PL
#model fits (models = 7)
#Do not show R^2
## Not run:
plot_fit(data = df,
         models = 7,
         startA = 0,
         startB = 1.5,
```

```
  startC = 500,
  startD = 92,
  startG = 1500,
  col = "#FF0000", #Color of observed data (points)
 col1 = "#FF00FF", #Predicted data color from model 1 (line). Model = 1 <- 5PL Model
col2 = "#0101DF",#Predicted data color from model 2 (line). Model = 2 <- Gompertz Model
 col3 = "#088A08",#Predicted data color from model 3 (line). Model = 3 <- 4PL Model
  axisX =  "Time (h)", #Title X-Axis
  axisY = "Carbon dioxide production (g/L)", #Title Y-Axis
  breaksX = seq(0,200,20), #X-Axis scale (positions). 0,20,40,60,80,...
  limitsX = c(0,200), #X-Axis Limits
  breaksY = seq(0,90,10),#Y-Axis scale (positions). 0,10,20,30,40,...
  limitsY = c(0,95), #Y-Axis Limits
  font = "serif",
  font.size = 14,
  legend.position = "top",
  show.R2 = FALSE) #Do not show R^2

## End(Not run)
```

---

pred *Get the model's predicted values*

Description

A function that, based on the observed data, the independent variable (e.g. time in h) and the dependent variable (e.g. CO_2 production in g L^{-1}), performs the modeling of the fermentation curve based on the chosen model(s) (**5PL**, **Gompertz**, or/and **4PL**).

From the analyzed data, this function will provide the predicted data for each evaluated fermentation curve.

Usage

```
pred(
  data,
  model,
  startA,
  startB,
  startC,
  startD,
  startG,
  save.xls = FALSE,
  dir.save,
  xls.name
)
```

Arguments

data	Data frame to be analyzed. The data frame must be in the following order: • **First**: All columns containing the independent variable (e.g. *time in hours*) • **Second**: All columns containing dependent variables (e.g. CO_2 *g* L^{-1} *production*) • **Header**: Columns must contain a header. If the treatment **ID** is in the header, this **ID** will be used to name the graphics PDF files for each analyzed curve.
model	Model or models to be adjusted: • **Model = 1**. 5PL Model. • **Model = 2**. Gompertz Model. • **Model = 3**. 4PL Model.
startA	Starting estimate of the value of A for model.
startB	Starting estimate of the value of B for model.
startC	Starting estimate of the value of C for model.
startD	Starting estimate of the value of D for model.
startG	Starting estimate of the value of G for model.
save.xls	If TRUE, an xlsx file containing the predicted values of each curve will be saved in the working directory. If it is FALSE, the xlsx file will not be saved.
dir.save	Directory path where the xlsx file is to be saved.
xls.name	File name. Must contain the format. For example, "Predicted Values.xlsx".

Details

Curve fitting from the observed data is performed by the nlsLM() function in the 'minpack.lm' package.

Value

The predicted values of each analyzed curve will be returned in a data.frame. In addition, a file **"Predicted Values.xlsx"** can be generated, containing the predicted values of each fermentation curve studied.

Author(s)

Angelo Gava

Examples

```
#Creating a data.frame.
#First, columns containing independent variable.
#Second, columns containing dependent variable.
#The data frame created presents two
#fermentation curves for two yeasts with
#different times and carbon dioxide production.
```

```
df <- data.frame('Time_Yeast_A' = seq(0,280, by=6.23),
                 'Time_Yeast_B' = seq(0,170, by=3.7777778),
                 'CO2_Production_Yeast_A' = c(0,0.97,4.04,9.62,13.44,17.50,
                                              24.03,27.46,33.75,36.40,40.80,
                                             44.24,48.01,50.85,54.85,57.51,
                                             61.73,65.43,66.50,72.41,75.47,
                                             77.22,78.49,79.26,80.31,81.04,
                                             81.89,82.28,82.56,83.13,83.62,
                                             84.11,84.47,85.02,85.31,85.61,
                                             86.05,86.27,85.29,86.81,86.94,
                                             87.13,87.33,87.45,87.85),
                 'CO2_Production_Yeast_B' = c(0,0.41,0.70,3.05,15.61,18.41,
                                              21.37,23.23,28.28,41.28,43.98,
                                              49.54,54.43,60.40,63.75,69.29,
                                              76.54,78.38,80.91,83.72,84.66,
                                              85.39,85.81,86.92,87.38,87.61,
                                              88.38,88.57,88.72,88.82,89.22,
                                              89.32,89.52,89.71,89.92,90.11,
                                              90.31,90.50,90.70,90.90,91.09,
                                              91.29,91.49,91.68,91.88))

#Using the pred() function to find the
#predicted valuesaccording to the adopted model.

pred(data = df,
model = 1,
startA = 0,
startB = 1.5,
startC = 500,
startD = 92,
startG = 1500,
save.xls = FALSE) #5PL Model adopted

pred(data = df,
model = 2,
startA = 92,
startB = 1.5,
startC = 0,
startD = NA,
startG = NA,
save.xls = FALSE) #Gompertz Model adopted

pred(data = df,
startA = 0,
startB = 2.5,
startC = 10,
startD = 92,
startG = NA,
model = 3,
save.xls = FALSE) #4PL Model adopted

#Saving an xlsx file. In this example,
```

```
#we will use saving a temporary file in
#the temporary file directories.
```

Index