Package ‘sicegar’

August 23, 2019

Type Package
Title Analysis of Single-Cell Viral Growth Curves
Version 0.2.3
Description Aims to quantify time intensity data by using sigmoidal and double sigmoidal curves. It fits straight lines, sigmoidal, and double sigmoidal curves on to time vs intensity data. Then all the fits are used to make decision on which model (sigmoidal, double sigmoidal, no signal or ambiguous) best describes the data. No signal means the intensity does not reach a high enough point or does not change at all over time. Sigmoidal means intensity starts from a small number than climbs to a maximum. Double sigmoidal means intensity starts from a small number, climbs to a maximum then starts to decay. After the decision between those four options, the algorithm gives the sigmoidal (or double sigmoidal) associated parameter values that quantifies the time intensity curve. The origin of the package name came from “Single Cell Growth Analysis in R”.
URL https://github.com/wilkelab/sicegar
Imports dplyr, minpack.lm, fBasics, ggplot2, stats
License GPL-2 | GPL-3
LazyData true
Suggests covr, cowplot, testthat, vdiffr, knitr
VignetteBuilder knitr
BugReports https://github.com/wilkelab/sicegar/issues
Collate 'categorize.R' 'mainFunctions.R' 'multipleFitFunction.R' 'sigmoidalFitFunctions.R' 'doublesigmoidalFitFunctions.R' 'normalizationFunction.R' 'sicegar.R' 'dataInputCheck.R' 'parameterCalculation.R' 'figureGeneration.R'
RoxygenNote 6.1.1
NeedsCompilation no
Author M. Umut Caglar [aut],
  Claus O. Wilke [aut, cre] (<https://orcid.org/0000-0002-7470-9261>)
Maintainer Claus O. Wilke <wilke@austin.utexas.edu>
Repository CRAN
Date/Publication 2019-08-23 15:30:02 UTC
**Description**

Categorizes the input data using the results of two model fits and chosen thresholds.

**Usage**

```r
categorize(parameterVectorSigmoidal, parameterVectorDoubleSigmoidal,  
threshold_intensity_range = 0.1,  
threshold_minimum_for_intensity_maximum = 0.3,  
threshold_bonus_sigmoidal_AIC = 0,  
threshold_sm_tmax_IntensityRatio = 0.85,  
threshold_dsm_tmax_IntensityRatio = 0.75, threshold_AIC = -10,  
threshold_t0_max_int = 0.05, showDetails = FALSE)
```

**Arguments**

- `parameterVectorSigmoidal`  
  Output of the sigmoidalFitFunction.

- `parameterVectorDoubleSigmoidal`  
  Output of the doublesigmoidalFitFunction.

- `threshold_intensity_range`  
  Minimum for intensity range, i.e. it is the lower limit for the allowed difference between the maximum and minimum of the intensities (Default is 0.1, and the values are based on actual, not the rescaled data.).
threshold_minimum_for_intensity_maximum
Minimum allowed value for intensity maximum. (Default is 0.3, and the values are based on actual, not the rescaled data.).

threshold_bonus_sigmoidal_AIC
Bonus AIC points for sigmoidal fit. Negative values help the sigmoidal model to win. Only helps in competition between sigmoidal and double sigmoidal fit at decision step "9", i.e. if none of the models fail in any of the tests and stay as a candidate until the last step (Default is 0).

threshold_sm_tmax_IntensityRatio
The threshold for the minimum intensity ratio between the last observed time points intensity and theoretical maximum intensity of the sigmoidal curve. If the value is below the threshold, then the data can not be represented with the sigmoidal model. (Default is 0.85)

threshold_dsm_tmax_IntensityRatio
The threshold for the minimum intensity ratio between the last observed time points intensity and maximum intensity of the double sigmoidal curve. If the value is above the threshold, then the data can not be represented with the double sigmoidal model. (Default is 0.75)

threshold_AIC
Maximum AIC values in order to have a meaningful fit (Default is -10).

threshold_t0_max_int
Maximum allowed intensity of the fitted curve at time is equal to zero (t=0). (Default is 0.05, and the values are based on actual, not the rescaled data.).

showDetails
Logical to chose if we want to see details or not. Default is "FALSE"

Value
The returned object contains extensive information about the decision process, but the key component is the decision variable. The decision variable can be one of the following four; "no_signal", "infection", "infection&lysis" or "ambiguous".

Examples
# Example 1 with double sigmoidal data
time=seq(3, 24, 0.1)

#simulate intensity data and add noise
noise_parameter <- 0.2
intensity_noise <- runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- sicegar::doublesigmoidalFitFormula(time,
                                                finalAsymptoteIntensityRatio = .3,
                                                maximum = 4,
                                                slope1Param = 1,
                                                midPoint1Param = 7,
                                                slope2Param = 1,
                                                midPointDistanceParam = 8)

intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- sicegar::normalizeData(dataInput,
# Fit sigmoidal model
sigmoidalModel <- sicegar::multipleFitFunction(dataInput = normalizedInput,
  model = "sigmoidal",
  n_runs_min = 20,
  n_runs_max = 500,
  showDetails = FALSE)

# Fit double sigmoidal model
doubleSigmoidalModel <- sicegar::multipleFitFunction(dataInput = normalizedInput,
  model = "doublesigmoidal",
  n_runs_min = 20,
  n_runs_max = 500,
  showDetails = FALSE)

# Calculate additional parameters
sigmoidalModel <- sicegar::parameterCalculation(sigmoidalModel)
doubleSigmoidalModel <- sicegar::parameterCalculation(doubleSigmoidalModel)

outputCluster <- sicegar::categorize(parameterVectorSigmoidal = sigmoidalModel,
  parameterVectorDoubleSigmoidal = doubleSigmoidalModel)

utils::str(outputCluster)

---

dataCheck

Checks if data is in correct format.

Description

Checks if the input data is appropriate and if it is not, the function converts it into a suitable form. The input data frame should contain two columns named time and intensity related to time variable and intensity variable respectively. If the data frame is in a list its name in the list should be $timeIntensityData.

Usage

dataCheck(data, showDetails = TRUE)

Arguments

data the input data. It can be either a list that contains a data frame in $timeIntensityData or can be a data frame by itself.

showDetails logical, if TRUE the function will provide an output "check done" if everything is OK. Default is FALSE
Examples

# Example 1

# generate data frame
time <- seq(3, 48, 0.5)
intensity <- runif(length(time), 3.0, 7.5)
dataInput <- data.frame(time, intensity)

# Apply dataCheck function
dataOutputVariable <- dataCheck(dataInput)

# Example 2

# generate data frame
time <- seq(3, 48, 0.5)
intensity <- runif(length(time), 3.0, 7.5)
dataInput <- data.frame(time, intensity)

# Normalize Data
dataOutput <- normalizeData(dataInput)
dataInput2 <- dataOutput

# Apply dataCheck function
dataOutputVariable2 <- dataCheck(dataInput2)

doublesigmoidalFitFormula

Double Sigmoidal Formula

Description

Calculates intensities using the double-sigmoidal model fit and the parameters (maximum, final asymptote intensity, slope1Param, midpoint1Param, slope2Param, and midpoint distance).

Usage

doublesigmoidalFitFormula(x, finalAsymptoteIntensityRatio, maximum, slope1Param, midpoint1Param, slope2Param, midpointDistanceParam)

Arguments

x the "time" (time) column of the dataframe
finalAsymptoteIntensityRatio This is the ratio between asymptote intensity and maximum intensity of the fitted curve.
maximum the maximum intensity that the double sigmoidal function reach.
slope1Param  the slope parameter of the sigmoidal function at the stepppest point in the exponential phase of the viral production.

midPoint1Param  the x axis value of the stepppest point in the function.

slope2Param  the slope parameter of the sigmoidal function at the stepppest point in the lysis phase. i.e when the intensity is decreasing.

midPointDistanceParam  the distance between the time of stepppest increase and stepppest decrease in the intensity data. In other words the distance between the x axis values of arguments of slope1Param and slope2Param.

Value

Returns the predicted intensities for the given time points with the double-sigmoidal fitted parameters for the double sigmoidal fit.

Examples

time <- seq(3, 24, 0.1)

#simulate intensity data and add noise
noise_parameter <- 0.2
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- doublesigmoidalFitFormula(time,
    finalAsymptoteIntensityRatio = .3,
    maximum = 4,
    slope1Param = 1,
    midPoint1Param = 7,
    slope2Param = 1,
    midPointDistanceParam = 8)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput)
parameterVector <- doublesigmoidalFitFunction(normalizedInput, tryCounter = 2)

#Check the results
if(parameterVector$isThisaFit){
    intensityTheoretical <-
    doublesigmoidalFitFormula(
        time,
        finalAsymptoteIntensityRatio = parameterVector$finalAsymptoteIntensityRatio_Estimate,
        maximum = parameterVector$maximum_Estimate,
        slope1Param = parameterVector$slope1Param_Estimate,
        midPoint1Param = parameterVector$midPoint1Param_Estimate,
        slope2Param = parameterVector$slope2Param_Estimate,
        midPointDistanceParam = parameterVector$midPointDistanceParam_Estimate)
    comparisonData <- cbind(dataInput, intensityTheoretical)

    require(ggplot2)
doublesigmoidalFitFunction

**Double sigmoidal fit function.**

**Description**

The function fits a double sigmoidal curve to given data by using likelihood maximization (LM) algorithm and provides the parameters (maximum, final asymptote intensity, slope1Param, midPoint1Param, slope2Param, and midPointDistance) describing the double-sigmoidal fit as output. It also contains information about the goodness of fits such as AIC, BIC, residual sum of squares, and log likelihood.

**Usage**

```r
doublesigmoidalFitFunction(dataInput, tryCounter, 
startList = list(finalAsymptoteIntensityRatio = 0, maximum = 1, 
slope1Param = 1, midPoint1Param = 0.33, slope2Param = 1, 
midPointDistanceParam = 0.29), 
lowerBounds = c(finalAsymptoteIntensityRatio = 0, maximum = 0.3, 
slope1Param = 0.01, midPoint1Param = -0.52, slope2Param = 0.01, 
midPointDistanceParam = 0.04), 
upperBounds = c(finalAsymptoteIntensityRatio = 1, maximum = 1.5, 
slope1Param = 180, midPoint1Param = 1.15, slope2Param = 180, 
midPointDistanceParam = 0.63), min_Factor = 1/2^20, 
n_iterations = 1000)
```

**Arguments**

- `dataInput`: A data frame or a list containing the dataframe. The data frame should be composed of at least two columns. One represents time, and the other represents intensity. The data should be normalized with the normalize data function `scigar::normalizeData()` before imported into this function.
- `tryCounter`: A counter that shows the number of times the data was fit via maximum likelihood function.
startList  The initial set of parameters vector that algorithm tries for the first fit attempt for the relevant parameters. The vector composes of six elements: 'finalAsymptoteIntensityRatio', 'maximum', 'slope1Param', 'midPoint1Param', 'slope2Param', and 'midPointDistanceParam'. Detailed explanations of those parameters can be found in vignettes. Defaults are finalAsymptoteIntensityRatio = 0, maximum = 1, slope1Param = 1, midPoint1Param = 0.33, slope2Param = 1, and midPointDistanceParam=0.29. The numbers are in normalized time intensity scale.

lowerBounds  The lower bounds for the randomly generated start parameters. The vector composes of six elements; 'finalAsymptoteIntensityRatio', 'maximum', 'slope1Param', 'midPoint1Param', 'slope2Param', and 'midPointDistanceParam'. Detailed explanations of those parameters can be found in vignettes. Defaults are finalAsymptoteIntensityRatio = 0, maximum = 0.3, slope1Param = .01, midPoint1Param = -0.52, slope2Param = .01, and midPointDistanceParam = 0.04. The numbers are in normalized time intensity scale.

upperBounds  The upper bounds for the randomly generated start parameters. The vector composes of six elements; 'finalAsymptoteIntensityRatio', 'maximum', 'slope1Param', 'midPoint1Param', 'slope2Param', and 'midPointDistanceParam'. Detailed explanations of those parameters can be found in vignettes. Defaults are finalAsymptoteIntensityRatio = 1, maximum = 1.5, slope1Param = 180, midPoint1Param = 1.15, slope2Param = 180, and midPointDistanceParam = 0.63. The numbers are in normalized time intensity scale.

min_Factor  Defines the minimum step size used by the fitting algorithm. Default is 1/2^20.

n_iterations  Define maximum number of iterations used by the fitting algorithm. Default is 1000

Value
Returns the fitted parameters and goodness of fit metrics.

Examples

time=seq(3, 24, 0.1)

# simulate intensity data and add noise
noise_parameter <- 0.2
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- doublesigmoidalFitFormula(time,
  finalAsymptoteIntensityRatio = .3,
  maximum = 4,
  slope1Param = 1,
  midPoint1Param = 7,
  slope2Param = 1,
  midPointDistanceParam = 8)

intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput)
parameterVector <- doublesigmoidalFitFunction(normalizedInput, tryCounter = 2)
# Check the results
if (parameterVector$isThisaFit)
{
  intensityTheoretical <-
  doublesigmoidalFitFormula(
    time,
    finalAsymptoteIntensityRatio = parameterVector$finalAsymptoteIntensityRatio_Estimate,
    maximum = parameterVector$maximum_Estimate,
    slope1Param = parameterVector$slope1Param_Estimate,
    midPoint1Param = parameterVector$midPoint1Param_Estimate,
    slope2Param = parameterVector$slope2Param_Estimate,
    midPointDistanceParam = parameterVector$midPointDistanceParam_Estimate
  )

  comparisonData <- cbind(dataInput, intensityTheoretical)

  require(ggplot2)
  ggplot(comparisonData) +
  geom_point(aes(x = time, y = intensity)) +
  geom_line(aes(x = time, y = intensityTheoretical)) +
  expand_limits(x = 0, y = 0)

  if (!parameterVector$isThisaFit) {print(parameterVector)}
}

figureModelCurves

Generate model associated figures.

Description
Generates figures using ggplot that shows the input data and the fitted curves.

Usage

figureModelCurves(dataInput, sigmoidalFitVector = NULL,
                   doubleSigmoidalFitVector = NULL, showParameterRelatedLines = FALSE,
                   xlabelText = "time", ylabelText = "intensity", fittedXmin = 0,
                   fittedXmax = NA)

Arguments

dataInput
  A data frame or a list containing the dataframe. The data frame should be composed of at least two columns. One represents time, and the other represents intensity. The data should be normalized with the normalize data function sicegar::normalizeData() before imported into this function.

sigmoidalFitVector
  the output of the sicegar::sigmoidalFitFunction(), or the augmented version of the output generated by the help of sicegar::parameterCalculation(), which contains parameters related with sigmoidal model. Default is NULL.
doubleSigmoidalFitVector
the output of the sicegar::doubleSigmoidalFitFunction(), or the augmented version of the output generated by the help of sicegar::parameterCalculation(), which contains parameters related with double sigmoidal model. Default is NULL.

theagumented version of the output generated by the help of sicegar::parameterCalculation(), which contains parameters related with double sigmoidal model. Default is NULL.

showParameterRelatedLines
if equal to TRUE, figure will show parameter related lines on the curves. Default is FALSE.

xlabelText
the x-axis name; with default "time"

ylabelText
the y-axis name; with default "intensity"

fittedXmin
the minimum of the fitted data that will be plotted (Default 0)

fittedXmax
the maximum of the fitted data that will be plotted (Default timeRange)

Value
Returns infection curve figures.

Examples

time <- seq(3, 24, 0.1)

#simulate intensity data and add noise
noise_parameter <- 0.2
intensity_noise <- runif(n = length(time), min = 0, max = 1) * noise_parameter

intensity <- sicegar::doublesigmoidalFitFormula(time, 
  finalAsymptoteIntensityRatio = .3, 
  maximum = 4, 
  slope1Param = 1, 
  midPoint1Param = 7, 
  slope2Param = 1, 
  midPointDistanceParam = 8)

intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)

normalizedInput <- sicegar::normalizeData(dataInput, dataInputName = "sample001")

# Do the double sigmoidal fit
doubleSigmoidalModel <- sicegar::multipleFitFunction(dataInput = normalizedInput, 
  model = "doublesigmoidal", 
  n_runs_min = 20, 
  n_runs_max = 500, 
  showDetails = FALSE)

doubleSigmoidalModel <- sicegar::parameterCalculation(doubleSigmoidalModel)

doubleSigmoidalModel <- sicegar::parameterCalculation(doubleSigmoidalModel)

doubleSigmoidalFitVector = doubleSigmoidalModel,
  showParameterRelatedLines = TRUE)

print(fig01)
fitAndCategorize

**Description**

Fits the sigmoidal and double-sigmoidal models to the data and then categorizes the data according to which model fits best.

**Usage**

```r
globalOptions

fitAndCategorize(dataInput, dataInputName = NA, n_runs_min_sm = 20, n_runs_max_sm = 500, n_runs_min_dsm = 20, n_runs_max_dsm = 500, showDetails = FALSE, startList_sm = list(maximum = 1, slopeParam = 1, midPoint = 0.33), lowerBounds_sm = c(maximum = 0.3, slopeParam = 0.01, midPoint = -0.52), upperBounds_sm = c(maximum = 1.5, slopeParam = 180, midPoint = 1.15), min_Factor_sm = 1/2^20, n_iterations_sm = 1000, startList_dsm = list(finalAsymptoteIntensityRatio = 0, maximum = 1, slope1Param = 1, midPoint1Param = 0.33, slope2Param = 1, midPointDistanceParam = 0.29), lowerBounds_dsm = c(finalAsymptoteIntensityRatio = 0, maximum = 0.3, slope1Param = 0.01, midPoint1Param = -0.52, slope2Param = 0.01, midPointDistanceParam = 0.04), upperBounds_dsm = c(finalAsymptoteIntensityRatio = 1, maximum = 1.5, slope1Param = 180, midPoint1Param = 1.15, slope2Param = 180, midPointDistanceParam = 0.63), min_Factor_dsm = 1/2^20, n_iterations_dsm = 1000, threshold_intensity_range = 0.1, threshold_minimum_for_intensity_maximum = 0.3, threshold_bonus_sigmoidal_AIC = 0, threshold_sm_tmax_IntensityRatio = 0.85, threshold_dsm_tmax_IntensityRatio = 0.75, threshold_AIC = -10, threshold_t0_max_int = 0.05, stepSize = 1e-05, ...)
```

**Arguments**

- `dataInput` - Un-normalized input data that will be fitted transferred into related functions
- `dataInputName` - Name of data set (Default is 'NA').
- `n_runs_min_sm` - This number indicates the lower limit of the successful fitting attempts for sigmoidal model. It should be smaller than the upper limit of the fitting attempts (n_runs_max_sm). Default is 20
- `n_runs_max_sm` - This number indicates the upper limit of the fitting attempts for sigmoidal model. Default is 500
- `n_runs_min_dsm` - This number indicates the lower limit of the successful fitting attempts for double sigmoidal model. It should be smaller than the upper limit of the fitting attempts (n_runs_max_dsm). Default is 20
- `n_runs_max_dsm` - This number indicates the upper limit of the fitting attempts for sigmoidal model for double sigmoidal model. Default is 500
showDetails Logical if TRUE prints details of intermediate steps of individual fits (Default is FALSE).

startList_sm The initial set of parameters vector that sigmoidal fit algorithm tries for the first fit attempt for the relevant parameters. The vector composes of three elements; 'maximum', 'slopeParam' and, 'midPoint'. Detailed explanations of those parameters can be found in vignettes. Defaults are maximum = 1, slopeParam = 1 and, midPoint = 0.33. The numbers are in normalized time intensity scale.

lowerBounds_sm The lower bounds for the randomly generated start parameters for the sigmoidal fit. The vector composes of three elements; 'maximum', 'slopeParam' and, 'midPoint'. Detailed explanations of those parameters can be found in vignettes. Defaults are maximum = 0.3, slopeParam = 0.01, and midPoint = -0.52. The numbers are in normalized time intensity scale.

upperBounds_sm The upper bounds for the randomly generated start parameters for the sigmoidal fit. The vector composes of three elements; 'maximum', 'slopeParam' and, 'midPoint'. Detailed explanations of those parameters can be found in vignettes. Defaults are maximum = 1.5, slopeParam = 180, midPoint = 1.15. The numbers are in normalized time intensity scale.

min_Factor_sm Defines the minimum step size used by the sigmoidal fit algorithm. Default is 1/2^20.

n_iterations_sm Defines maximum number of iterations used by the sigmoidal fit algorithm. Default is 1000

startList_dsm The initial set of parameters vector that double sigmoidal fit algorithm tries for the first fit attempt for the relevant parameters. The vector composes of six elements; 'finalAsymptoteIntensityRatio', 'maximum', 'slope1Param', 'midPoint1Param', 'slope2Param', and 'midPointDistanceParam'. Detailed explanations of those parameters can be found in vignettes. Defaults are finalAsymptoteIntensityRatio = 0, maximum = 1, slope1Param = 1, midPoint1Param = 0.33, slope2Param = 1, and midPointDistanceParam=0.29. The numbers are in normalized time intensity scale.

lowerBounds_dsm The lower bounds for the randomly generated start parameters for double sigmoidal fit. The vector composes of six elements; 'finalAsymptoteIntensityRatio', 'maximum', 'slope1Param', 'midPoint1Param', 'slope2Param', and 'midPointDistanceParam'. Detailed explanations of those parameters can be found in vignettes. Defaults are finalAsymptoteIntensityRatio = 0, maximum = 0.3, slope1Param = .01, midPoint1Param = -0.52, slope2Param = .01, and midPointDistanceParam = 0.04. The numbers are in normalized time intensity scale.

upperBounds_dsm The upper bounds for the randomly generated start parameters for double sigmoidal fit. The vector composes of six elements; 'finalAsymptoteIntensityRatio', 'maximum', 'slope1Param', 'midPoint1Param', 'slope2Param', and 'midPointDistanceParam'. Detailed explanations of those parameters can be found in vignettes. Defaults are finalAsymptoteIntensityRatio = 1, maximum = 1.5, slope1Param = 180, midPoint1Param = 1.15, slope2Param = 180, and midPointDistanceParam = 0.63. The numbers are in normalized time intensity scale.
min_Factor_dsm  Defines the minimum step size used by the double sigmoidal fit algorithm. Default is 1/2^20.

n_iterations_dsm Define maximum number of iterations used by the double sigmoidal fit algorithm. Default is 1000

threshold_intensity_range Minimum for intensity range, i.e. it is the lower limit for the allowed difference between the maximum and minimum of the intensities (Default is 0.1, and the values are based on actual, not the rescaled data.).

threshold_minimum_for_intensity_maximum Minimum allowed value for intensity maximum. (Default is 0.3, and the values are based on actual, not the rescaled data.).

threshold_bonus_sigmoidal_AIC Bonus AIC points for sigmoidal fit. Negative values help the sigmoidal model to win. Only helps in competition between sigmoidal and double sigmoidal fit at decision step "9", i.e. if none of the models fail in any of the tests and stay as a candidate until the last step (Default is 0).

threshold_sm_tmax_IntensityRatio The threshold for the minimum intensity ratio between the last observed time points intensity and theoretical maximum intensity of the sigmoidal curve. If the value is below the threshold, then the data can not be represented with the sigmoidal model. (Default is 0.85)

threshold_dsm_tmax_IntensityRatio The threshold for the minimum intensity ratio between the last observed time points intensity and maximum intensity of the double sigmoidal curve. If the value is above the threshold, then the data can not be represented with the double sigmoidal model. (Default is 0.75)

threshold_AIC Maximum AIC values in order to have a meaningful fit (Default is -10).

threshold_t0_max_int Maximum allowed intensity of the fitted curve at time is equal to zero (t=0). (Default is 0.05, and the values are based on actual, not the rescaled data.).

stepSize Step size used by the fitting algorithm. Smaller numbers gave more accurate results than larger numbers, and larger numbers gave the results faster than small numbers. The default value is 0.00001.

... All other arguments that model functions ("sigmoidalFitFunction" and, "double-sigmoidalFitFunction") may need.

Value

Returns the parameters related with the curve fitted to the input data.

Examples

# Example 1
time <- seq(3, 24, 0.5)

#simulate intensity data and add noise
multipleFitFunction

multiple fit function.

Description

Calls the fitting algorithms to fit the data multiple times with starting from different randomly generated initial parameters in each run. Multiple attempts at fitting the data are necessary to avoid local minima.

Usage

multipleFitFunction(dataInput, dataInputName = NA, model, n_runs_min = 20, n_runs_max = 500, showDetails = FALSE, ...)

Arguments

dataInput A data frame or a list containing the dataframe. The data frame should be composed of at least two columns. One represents time, and the other represents intensity. The data should be normalized with the normalize data function sicegar::normalizeData() before imported into this function.

dataInputName Name of data set (Default is 'NA').

model Type of fit model that will be used. Can be "sigmoidal", or "double_sigmoidal".

n_runs_min This number indicates the lower limit of the successful fitting attempts. It should be smaller than the upper limit of the fitting attempts (n_runs_max). Default is 20.

n_runs_max This number indicates the upper limit of the fitting attempts. Default is 500.

showDetails Logical if TRUE prints details of intermediate steps of individual fits (Default is FALSE).

... All other arguments that model functions ("sigmoidalFitFunction" and, "double-sigmoidalFitFunction") may need.

noise_parameter <- 0.2
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- sicegar::doublesigmoidalFitFormula(time,
  finalAsymptoteIntensityRatio = .3,
  maximum = 4,
  slope1Param = 1,
  midPoint1Param = 7,
  slope2Param = 1,
  midPointDistanceParam = 8)

intensity <- intensity + intensity_noise
dataInput <- data.frame(intensity = intensity, time = time)
fitObj <- sicegar::fitAndCategorize(dataInput = dataInput)
**Value**

Returns the parameters related with the model fitted for the input data.

**Examples**

```r
# Example 1 (sigmoidal function with normalization)
# simulating intensity data and add noise
noise_parameter <- 2.5
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- sigmoidalFitFormula(time, maximum = 4, slopeParam = 1, midPoint = 8)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput, dataInputName = "sample001")
parameterVector <- multipleFitFunction(dataInput = normalizedInput,
model = "sigmoidal",
n_runs_min = 20,
n_runs_max = 500)

# Check the results
if(parameterVector$isThisaFit){
  intensityTheoretical <- sigmoidalFitFormula(time,
  maximum = parameterVector$maximum_Estimate,
  slopeParam = parameterVector$slopeParam_Estimate,
  midPoint = parameterVector$midPoint_Estimate)

  comparisonData <- cbind(dataInput, intensityTheoretical)
  print(parameterVector$residual_Sum_of_Squares)

  require(ggplot2)
  ggplot(comparisonData)+
  geom_point(aes(x = time, y = intensity)) +
  geom_line(aes(x = time, y = intensityTheoretical), color = "orange") +
  expand_limits(x = 0, y = 0)
}

if(!parameterVector$isThisaFit){
  print(parameterVector)
}

# Example 2 (doublesigmoidal function with normalization)
# simulating intensity data with noise
noise_parameter <- 0.2
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- doublesigmoidalFitFormula(time, maximum = 8, slopeParam1 = 1, midPoint1 = 8, slopeParam2 = 2, midPoint2 = 12)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput, dataInputName = "sample001")
parameterVector <- multipleFitFunction(dataInput = normalizedInput,
model = "doublesigmoidal",
n_runs_min = 20,
n_runs_max = 500)

# Check the results
if(parameterVector$isThisaFit){
  intensityTheoretical <- doublesigmoidalFitFormula(time,
  maximum1 = parameterVector$maximum_Estimate1,
  maximum2 = parameterVector$maximum_Estimate2,
  slopeParam1 = parameterVector$slopeParam_Estimate1,
  slopeParam2 = parameterVector$slopeParam_Estimate2,
  midPoint1 = parameterVector$midPoint_Estimate1,
  midPoint2 = parameterVector$midPoint_Estimate2)

  comparisonData <- cbind(dataInput, intensityTheoretical)
  print(parameterVector$residual_Sum_of_Squares)

  require(ggplot2)
  ggplot(comparisonData)+
  geom_point(aes(x = time, y = intensity)) +
  geom_line(aes(x = time, y = intensityTheoretical), color = "orange") +
  expand_limits(x = 0, y = 0)
}

if(!parameterVector$isThisaFit){
  print(parameterVector)
}
```

```r
# Example 3 (doublesigmoidal function with normalization)
# simulating intensity data with noise
noise_parameter <- 0.2
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- doublesigmoidalFitFormula(time,
maximum1 = 8, slopeParam1 = 1, midPoint1 = 8,
maximum2 = 12, slopeParam2 = 2, midPoint2 = 12)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput, dataInputName = "sample001")
parameterVector <- multipleFitFunction(dataInput = normalizedInput,
model = "doublesigmoidal",
n_runs_min = 20,
n_runs_max = 500)

# Check the results
if(parameterVector$isThisaFit){
  intensityTheoretical <- doublesigmoidalFitFormula(time,
  maximum1 = parameterVector$maximum_Estimate1,
  maximum2 = parameterVector$maximum_Estimate2,
  slopeParam1 = parameterVector$slopeParam_Estimate1,
  slopeParam2 = parameterVector$slopeParam_Estimate2,
  midPoint1 = parameterVector$midPoint_Estimate1,
  midPoint2 = parameterVector$midPoint_Estimate2)

  comparisonData <- cbind(dataInput, intensityTheoretical)
  print(parameterVector$residual_Sum_of_Squares)

  require(ggplot2)
  ggplot(comparisonData)+
  geom_point(aes(x = time, y = intensity)) +
  geom_line(aes(x = time, y = intensityTheoretical), color = "orange") +
  expand_limits(x = 0, y = 0)
}

if(!parameterVector$isThisaFit){
  print(parameterVector)
}
```

```r
# Example 4 (doublesigmoidal function with normalization)
# simulating intensity data with noise
noise_parameter <- 0.2
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- doublesigmoidalFitFormula(time,
maximum1 = 8, slopeParam1 = 1, midPoint1 = 8,
maximum2 = 12, slopeParam2 = 2, midPoint2 = 12)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput, dataInputName = "sample001")
parameterVector <- multipleFitFunction(dataInput = normalizedInput,
model = "doublesigmoidal",
n_runs_min = 20,
```
normalizeData

Normalization of given data

```r
finalAsymptoteIntensityRatio = 0.3,
maximum = 4,
slope1Param = 1,
midPoint1Param = 7,
slope2Param = 1,
midPointDistanceParam = 8)

intensity <- intensity + intensity_noise
dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput)

parameterVector <- multipleFitFunction(dataInput = normalizedInput,
dataInputName="sample001",
model = "doublesigmoidal",
n_runs_min = 20,
n_runs_max = 500,
showDetails = FALSE)

#Check the results
if(parameterVector$isThisaFit){
  intensityTheoretical <-
    doublesigmoidalFitFormula(
      time,
      finalAsymptoteIntensityRatio = parameterVector$finalAsymptoteIntensityRatio_Estimate,
      maximum = parameterVector$maximum_Estimate,
      slope1Param = parameterVector$slope1Param_Estimate,
      midPoint1Param = parameterVector$midPoint1Param_Estimate,
      slope2Param = parameterVector$slope2Param_Estimate,
      midPointDistanceParam = parameterVector$midPointDistanceParam_Estimate)

  comparisonData <- cbind(dataInput, intensityTheoretical)

  require(ggplot2)
  ggplot(comparisonData) +
  geom_point(aes(x = time, y = intensity)) +
  geom_line(aes(x = time, y = intensityTheoretical), color = "orange") +
  expand_limits(x = 0, y = 0)
}

if(!parameterVector$isThisaFit){
  print(parameterVector)
}
```
**Description**

Maps the given time-intensity data into a rescaled data frame where time is scaled in a way that maximum time point is one and intensity is distributed between [0,1].

**Usage**

normalizeData(dataInput, dataInputName = NA)

**Arguments**

dataInput A data frame or a list containing the dataframe. The data frame should be composed of at least two columns. One represents time, and the other represents intensity.
dataInputName experiment name (Default is 'NA').

**Value**

Function returns a new data frame, scaling factors and scaling constants that connects initial data frame to new one. The new data frame includes 2 columns one is for normalized time and the other is for noralized intensity. The whole time is distributed between 0 and 1 and similarly the whole intensity is distributed between 0 and 1. The time and intensity constants and scaling factors are the parameters to transform data from unnormalized data frame to normalized data frame.

**Examples**

```r
# generateRandomData
time <- seq(3, 48, 0.5)
intensity <- runif(length(time), 3.0, 7.5)
dataInput <- data.frame(time, intensity)

# Normalize Data
dataOutput <- normalizeData(dataInput, dataInputName="sample001")
```

---

**parameterCalculation**  
*useful parameter calculation with help of fits*

**Description**

Generates useful values for external use, with the help of parameterVector's of the fits.

**Usage**

parameterCalculation(parameterVector, stepSize = 1e-05)
Arguments

```r
parameterVector
Output of multiple fit function `sicegar::multipleFitFunction()` that gives the variables related with sigmoidal or double sigmoidal fit.
```

```r
stepSize
Step size used by the fitting algorithm. Smaller numbers gave more accurate results than larger numbers, and larger numbers gave the results faster than small numbers. The default value is 0.00001.
```

Value

Returns the expanded parameter vector. This vector includes useful derived values such as time and intensity of the start point, in addition to the standard values that the fit algorithms produce that are necessary to define the curves.

Examples

```r
time <- seq(3, 24, 0.1)

#simulate intensity data with noise
noise_parameter <- 0.2
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- sicegar::doublesigmoidalFitFormula(time,
   finalAsymptoteIntensityRatio = .3,
   maximum = 4,
   slope1Param = 1,
   midPoint1Param = 7,
   slope2Param = 1,
   midPointDistanceParam = 8)

intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- sicegar::normalizeData(dataInput)
parameterVector <- sicegar::multipleFitFunction(dataInput = normalizedInput,
   dataInputName = "sample01",
   model = "doublesigmoidal",
   n_runs_min = 20,
   n_runs_max = 500,
   showDetails = FALSE)

if(parameterVector$isThisaFit){
   parameterVector <- sicegar::parameterCalculation(parameterVector)
}

print(t(parameterVector))
```
Description

Checks if the signal is present in the data. Often a high percentage of high through-put data does not contain a signal. Checking if data does not contain signal before doing a sigmoidal or double sigmoidal fit can make the analysis of data from high-throughput experiments much faster.

Usage

```r
preCategorize(normalizedInput, threshold_intensity_range = 0.1,
threshold_minimum_for_intensity_maximum = 0.3)
```

Arguments

- `normalizedInput` is the output of the `sicegar::normalizeData()` function.
- `threshold_intensity_range` minimum for intensity range, i.e. it is the lower limit for the allowed difference between the maximum and minimum of the intensities (Default is 0.1, and the values are based on actual, not the rescaled data.).
- `threshold_minimum_for_intensity_maximum` minimum allowed value for intensity maximum. (Default is 0.3, and the values are based on actual, not the rescaled data.).

Value

Function returns a brief decision list that includes information about the decision process. Post important part of this information is `decisionList$decision` which might be either "no_signal" or "not_no_signal".

Examples

```r
# Example 1 with double sigmoidal data

time=seq(3, 24, 0.1)

#simulate intensity data and add noise
noise_parameter = 0.2
intensity_noise = runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity = sicegar::doublesigmoidalFitFormula(time,
finalAsymptoteIntensityRatio = .3,
maximum = 4,
slope1Param = 1,
midPoint1Param = 7,
slope2Param = 1,
midPointDistanceParam = 8)
```
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- sicegar::normalizeData(dataInput, dataInputName = "sample001")
isThis_nosignal <- sicegar::preCategorize(normalizedInput = normalizedInput)

# Example 2 with no_signal data

time <- seq(3, 24, 0.1)

# simulate intensity data and add noise
noise_parameter <- 0.05
intensity_noise <- runif(n = length(time), min = 0, max = 1) * noise_parameter * 2e-04
intensity <- sicegar::doublesigmoidalFitFormula(time,
  finalAsymptoteIntensityRatio = .3,
  maximum = 2e-04,
  slope1Param = 1,
  midPoint1Param = 7,
  slope2Param = 1,
  midPointDistanceParam = 8)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity=intensity, time=time)
normalizedInput <- sicegar::normalizeData(dataInput, dataInputName = "sample001")
isThis_nosignal <- sicegar::preCategorize(normalizedInput = normalizedInput)

---

sameSourceDataCheck

Check is data came from the same source.

Description

Checks if the provided data and models came from same source by looking to "dataInputName" columns of the inputs.

Usage

sameSourceDataCheck(dataInput, sigmoidalFitVector, doubleSigmoidalFitVector)

Arguments

dataInput a data frame composed of two columns. One is for time and the other is for intensity. Should be normalized data generated by normalizeData.
sigmoidalFitVector is the output of sigmoidalFitFunction. Default is NULL.
doubleSigmoidalFitVector is the output of double sigmoidal fit function. Default is NULL.
Value

Returns TRUE if models can from same source, FALSE otherwise.

Description

Calculates intensities for given time points (x) by using sigmoidal fit model and parameters (maximum, slopeParam, and midpoint).

Usage

sigmoidalFitFormula(x, maximum, slopeParam, midPoint)

Arguments

- x: the "time" (time) column of the dataframe.
- maximum: the maximum intensity that the sigmoidal function can reach while time approaches infinity.
- slopeParam: the slope parameter of the sigmoidal function at the steppest point.
- midPoint: the x axis value of the steppest point in the function.

Value

Returns the predicted intensities for given time points with the given sigmoidal fit parameters.

Examples

```r
time <- seq(3, 24, 0.5)
#simulate intensity data and add noise
noise_parameter <- 0.1
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- sigmoidalFitFormula(time, maximum = 4, slopeParam = 1, midPoint = 8)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput)
parameterVector <- sigmoidalFitFunction(normalizedInput, tryCounter = 2)

#Check the results
if(parameterVector$isThisaFit){
  intensityTheoretical <- sigmoidalFitFormula(time, maximum = parameterVector$maximum_Estimate,
                                              slopeParam = parameterVector$slopeParam_Estimate,
                                              midPoint = parameterVector$midPoint_Estimate)
}
```
sigmoidalFitFunction

sigmoidalFitFunction  Sigmoidal fit function

Description

The function fits a sigmoidal curve to given data by using likelihood maximization (LM) algorithm and provides the parameters (maximum, slopeParam and, midPoint) describing the double-sigmoidal fit as output. It also contains information about the goodness of fits such as AIC, BIC, residual sum of squares, and log likelihood.

Usage

sigmoidalFitFunction(dataInput, tryCounter, startList = list(maximum = 1, slopeParam = 1, midPoint = 0.33), lowerBounds = c(maximum = 0.3, slopeParam = 0.01, midPoint = -0.52), upperBounds = c(maximum = 1.5, slopeParam = 180, midPoint = 1.15), min_Factor = 1/2^20, n_iterations = 1000)

Arguments

dataInput  A data frame or a list containing the dataframe. The data frame should be composed of at least two columns. One represents time, and the other represents intensity. The data should be normalized with the normalize data function sicegar::normalizeData() before imported into this function.

tryCounter  A counter that shows the number of times the data was fit via maximum likelihood function.

startList  The initial set of parameters vector that algorithm tries for the first fit attempt for the relevant parameters. The vector composes of three elements; ‘maximum’, ‘slopeParam’ and, ‘midPoint’. Detailed explanations of those parameters can be found in vignettes. Defaults are maximum = 1, slopeParam = 1 and, midPoint = 0.33. The numbers are in normalized time intensity scale.
lowerBounds

The lower bounds for the randomly generated start parameters. The vector composes of three elements; 'maximum', 'slopeParam' and, 'midPoint'. Detailed explanations of those parameters can be found in vignettes. Defaults are maximum = 0.3, slopeParam = 0.01, and midPoint = -0.52. The numbers are in normalized time intensity scale.

upperBounds

The upper bounds for the randomly generated start parameters. The vector composes of three elements; 'maximum', 'slopeParam' and, 'midPoint'. Detailed explanations of those parameters can be found in vignettes. Defaults are maximum = 1.5, slopeParam = 180, midPoint = 1.15. The numbers are in normalized time intensity scale.

min_Factor

Defines the minimum step size used by the fitting algorithm. Default is 1/2^20.

n_iterations

Defines maximum number of iterations used by the fitting algorithm. Default is 1000

Value

Returns fitted parameters for the sigmoidal model.

Examples

time <- seq(3, 24, 0.5)
#simulate intensity data and add noise
noise_parameter <- 0.1
intensity_noise <- stats::runif(n = length(time), min = 0, max = 1) * noise_parameter
intensity <- sigmoidalFitFormula(time, maximum = 4, slopeParam = 1, midPoint = 8)
intensity <- intensity + intensity_noise

dataInput <- data.frame(intensity = intensity, time = time)
normalizedInput <- normalizeData(dataInput)
parameterVector <- sigmoidalFitFunction(normalizedInput, tryCounter = 2)

#Check the results
if(parameterVector$isThisaFit){
    intensityTheoretical <- sigmoidalFitFormula(time,
        maximum = parameterVector$maximum_Estimate,
        slopeParam = parameterVector$slopeParam_Estimate,
        midPoint = parameterVector$midPoint_Estimate)
    comparisonData <- cbind(dataInput, intensityTheoretical)

    require(ggplot2)
    ggplot(comparisonData) +
    geom_point(aes(x = time, y = intensity)) +
    geom_line(aes(x = time, y = intensityTheoretical)) +
    expand_limits(x = 0, y = 0)
}

if(!parameterVector$isThisaFit){
    print(parameterVector)
}
unnormalizeData

Unnormalization of given data

Description
Maps the given time-intensity data into a rescaled frame where time is between [0,1] and similarly intensity is between [0,1].

Usage
unnormalizeData(dataInput)

Arguments
dataInput  a list file composes of two parts First part is the data that will be unnormalized, which is a data frame composed of two columns. One is for time and the other is for intensity Second part is the scaling parameters of the data which is a vector that has three components. The first one of them is related with time and last two of them are related with intensity. The second value represents the min value of the intensity set. First and third values represent the difference between max and min value in the relevant column.

Value
Returns a data frame, scaling factors and scaling constants for time and intensity. The other data frame includes 2 columns one is for normalized time and the other is for normalized intensity. The whole time is distributed between 0 and 1 and similarly the whole intensity is distributed between 0 and 1. The time and intensity constants and scaling factors are the parameters to transform data from given set to scaled set.

Examples
# generateRandomData
time <- seq(3, 48, 0.5)
intensity <- runif(length(time), 3.0, 7.5)
dataInput <- data.frame(time, intensity)
# Normalize Data
dataOutput <- normalizeData(dataInput)
dataInput2 <- dataOutput
# Un Normalize it
dataOutput2 <- unnormalizeData(dataInput2)
Index

categorize, 2

dataCheck, 4
doublesigmoidalFitFormula, 5
doublesigmoidalFitFunction, 7

figureModelCurves, 9
fitAndCategorize, 11

multipleFitFunction, 14

normalizeData, 16

parameterCalculation, 17
preCategorize, 19

sameSourceDataCheck, 20
sigmoidalFitFormula, 21
sigmoidalFitFunction, 22

unnormalizeData, 24

25