Package ‘CGNM’

June 12, 2024

Type Package

Title Cluster Gauss-Newton Method

Version 0.9.0

Author Yasunori Aoki

Maintainer Yasunori Aoki <`yaoki@uwaterloo.ca`>

Description Find multiple solutions of a nonlinear least squares problem. Cluster Gauss-Newton method does not assume uniqueness of the solution of the nonlinear least squares problem and compute multiple minimizers. Please cite the following paper when this software is used in your research: Aoki et al. (2020) <doi:10.1007/s11081-020-09571-2>. Cluster Gauss–Newton method. Optimization and Engineering, 1-31. Please cite the following paper when profile likelihood plot is drawn with this software and used in your research: Aoki and Sugiyama (2024) <doi:10.1002/psp4.13055>. Cluster Gauss-Newton method for a quick approximation of profile likelihood: With application to physiologically-based pharmacokinetic models. CPT Pharmacometrics Syst Pharmacol.13(1):54-67.

License MIT + file LICENSE

Encoding UTF-8

Imports stats, ggplot2, MASS, shiny

Suggests knitr, rmarkdown

RoxygenNote 7.3.1

VignetteBuilder knitr

NeedsCompilation no

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-06-12 05:50:02 UTC

Contents

`acceptedApproximateMinimizers`

acceptedApproximateMinimizers

Description

CGNM find multiple sets of minimizers of the nonlinear least squares (nls) problem by solving nls from various initial iterates. Although CGNM is shown to be robust compared to other conventional multi-start algorithms, not all initial iterates minimizes successfully. By assuming sum of squares residual (SSR) follows the chai-square distribution we first reject the approximated minimiser who SSR is statistically significantly worse than the minimum SSR found by the CGNM. Then use elbow-method (a heuristic often used in mathematical optimisation to balance the quality and the quantity of the solution found) to find the "acceptable" maximum SSR. This function outputs the acceptable approximate minimizers of the nonlinear least squares problem found by the CGNM.

Usage

```
acceptedApproximateMinimizers(
  CGNM_result,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  useAcceptedApproximateMinimizers = TRUE,
  algorithm = 2,
  ParameterNames = NA,
```

```
  ReparameterizationDef = NA
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`cutoff_pvalue`	(default: 0.05) *A number* defines the rejection p-value for the first stage of acceptable computational result screening.
`numParametersIncluded`	(default: NA) *A natural number* defines the number of parameter sets to be included in the assessment of the acceptable parameters. If set NA then use all the parameters found by the CGNM.
`useAcceptedApproximateMinimizers`	(default: TRUE) *TRUE or FALSE* If true then use chai-square and elbow method to choose maximum accepted SSR. If false returns the parameters upto numParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then use all the parameters found by the CGNM).
`algorithm`	(default: 2) *1 or 2* specify the algorithm used for obtain accepted approximate minimizers. (Algorithm 1 uses elbow method, Algorithm 2 uses Grubbs' Test for Outliers.)
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax

Value

A dataframe that each row stores the accepted approximate minimizers found by CGNM.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}
```

```
observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

acceptedApproximateMinimizers(CGNM_result)
```

`acceptedIndices` *acceptedIndices*

Description

CGNM find multiple sets of minimizers of the nonlinear least squares (nls) problem by solving nls from various initial iterates. Although CGNM is shown to be robust compared to other conventional multi-start algorithms, not all initial iterates minimizes successfully. By assuming sum of squares residual (SSR) follows the chai-square distribution we first reject the approximated minimiser who SSR is statistically significantly worse than the minimum SSR found by the CGNM. Then use elbow-method (a heuristic often used in mathematical optimisation to balance the quality and the quantity of the solution found) to find the "acceptable" maximum SSR. This function outputs the indices of acceptable approximate minimizers of the nonlinear least squares problem found by the CGNM.

Usage

```
acceptedIndices(
  CGNM_result,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  useAcceptedApproximateMinimizers = TRUE,
  algorithm = 2
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`cutoff_pvalue`	(default: 0.05) *A number* defines the rejection p-value for the first stage of acceptable computational result screening.
`numParametersIncluded`	(default: NA) *A natural number* defines the number of parameter sets to be included in the assessment of the acceptable parameters. If set NA then use all the parameters found by the CGNM.

`useAcceptedApproximateMinimizers`	(default: TRUE) *TRUE or FALSE* If true then use chai-square and elbow method to choose maximum accepted SSR. If false returns the parameters upto numParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then use all the parameters found by the CGNM).
`algorithm`	(default: 2) *1 or 2* specify the algorithm used for obtain accepted approximate minimizers. (Algorithm 1 uses elbow method, Algorithm 2 uses Grubbs' Test for Outliers.)

Value

A vector of natural number that contains the indices of accepted approximate minimizers found by CGNM.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

acceptedIndices(CGNM_result)
```

`acceptedIndices_binary`

acceptedIndices_binary

Description

CGNM find multiple sets of minimizers of the nonlinear least squares (nls) problem by solving nls from various initial iterates. Although CGNM is shown to be robust compared to other conventional multi-start algorithms, not all initial iterates minimizes successfully. By assuming sum of squares residual (SSR) follows the chai-square distribution we first reject the approximated minimiser who SSR is statistically significantly worse than the minimum SSR found by the CGNM. Then use elbow-method (a heuristic often used in mathematical optimisation to balance the quality and the quantity of the solution found) to find the "acceptable" maximum SSR. This function outputs the indices of acceptable approximate minimizers of the nonlinear least squares problem found by the CGNM. (note that acceptedIndices(CGNM_result) is equal to seq(1,length(acceptedIndices_binary(CGNM_result)))[accepte

Usage

```
acceptedIndices_binary(
  CGNM_result,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  useAcceptedApproximateMinimizers = TRUE,
  algorithm = 2
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`cutoff_pvalue`	(default: 0.05) *A number* defines the rejection p-value for the first stage of acceptable computational result screening.
`numParametersIncluded`	(default: NA) *A natural number* defines the number of parameter sets to be included in the assessment of the acceptable parameters. If set NA then use all the parameters found by the CGNM.
`useAcceptedApproximateMinimizers`	(default: TRUE) *TRUE or FALSE* If true then use chai-square and elbow method to choose maximum accepted SSR. If false returns the indicies upto numParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then use all the parameters found by the CGNM).
`algorithm`	(default: 2) *1 or 2* specify the algorithm used for obtain accepted approximate minimizers. (Algorithm 1 uses elbow method, Algorithm 2 uses Grubbs' Test for Outliers.)

Value

A vector of TRUE and FALSE that indicate if the each of the approximate minimizer found by CGNM is acceptable or not.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

acceptedIndices_binary(CGNM_result)
```

`acceptedMaxSSR` *acceptedMaxSSR*

Description

CGNM find multiple sets of minimizers of the nonlinear least squares (nls) problem by solving nls from various initial iterates. Although CGNM is shown to be robust compared to other conventional multi-start algorithms, not all initial iterates minimizes successfully. By assuming sum of squares residual (SSR) follows the chai-square distribution we first reject the approximated minimiser who SSR is statistically significantly worse than the minimum SSR found by the CGNM. Then use elbow-method (a heuristic often used in mathematical optimisation to balance the quality and the quantity of the solution found) to find the "acceptable" maximum SSR.

Usage

```
acceptedMaxSSR(
  CGNM_result,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  useAcceptedApproximateMinimizers = TRUE,
  algorithm = 2
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`cutoff_pvalue`	(default: 0.05) *A number* defines the rejection p-value for the first stage of acceptable computational result screening.
`numParametersIncluded`	(default: NA) *A natural number* defines the number of parameter sets to be included in the assessment of the acceptable parameters. If set NA then use all the parameters found by the CGNM.
`useAcceptedApproximateMinimizers`	(default: TRUE) *TRUE or FALSE* If true then use chai-square and elbow method to choose maximum accepted SSR. If false returnsnumParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then returns the largest SSR).
`algorithm`	(default: 2) *1 or 2* specify the algorithm used for obtain accepted approximate minimizers. (Algorithm 1 uses elbow method, Algorithm 2 uses Grubbs' Test for Outliers.)

Value

A positive real number that is the maximum sum of squares residual (SSR) the algorithm has selected to accept.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

acceptedMaxSSR(CGNM_result)
```

`bestApproximateMinimizers`

bestApproximateMinimizers

Description

Returns the approximate minimizers with minimum SSR found by CGNM.

Usage

```
bestApproximateMinimizers(
  CGNM_result,
  numParameterSet = 1,
  ParameterNames = NA,
  ReparameterizationDef = NA
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`numParameterSet`	(default 1) *A natural number* number of parameter sets to output (chosen from the smallest SSR to numParameterSet-th smallest SSR) .
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax

Value

A vector a vector of accepted approximate minimizers with minimum SSR found by CGNM.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time
```

```
 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

bestApproximateMinimizers(CGNM_result,10)
```

`Cluster_Gauss_Newton_Bootstrap_method`

Cluster_Gauss_Newton_Bootstrap_method

Description

Conduct residual resampling bootstrap analyses using CGNM.

Usage

```
Cluster_Gauss_Newton_Bootstrap_method(
  CGNM_result,
  nonlinearFunction,
  num_bootstrapSample = 200,
  indicesToUseAsInitialIterates = NA,
  bootstrapType = 1,
  ...
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`nonlinearFunction`	(required input) *A function with input of a vector x of real number of length n and output a vector y of real number of length m.* In the context of model fitting the nonlinearFunction is **the model**. Given the CGNM does not assume the uniqueness of the minimizer, m can be less than n. Also CGNM does not assume any particular form of the nonlinear function and also does not require the function to be continuously differentiable (see Appendix D of our publication for an example when this function is discontinuous).

`num_bootstrapSample`	(default: 200) *A positive integer* number of bootstrap samples to generate.
`indicesToUseAsInitialIterates`	(default: NA) *A vector of integers* indices to use for initial iterate of the bootstrap analyses. For CGNM bootstrap, we use the parameters found by CGNM as the initial iterates, here you can manually spccify which of the approximate minimizers that was found by CGNM (where the CGNM computation result is given as CGNM_result file) to use as initial iterates. (if NA, use indices chosen by the acceptedIndices() function with default setting).
`bootstrapType`	(default:1) *1 or 2* 1: residual resampling bootstrap method, 2: case sampling bootstrap method
`...`	Further arguments to be supplied to nonlinearFunction

Value

list of a matrix X, Y,residual_history, initialX, bootstrapX, bootstrapY as well as a list runSetting.

1. X, Y, residual_history, initialX: identical to what was given as CGNM_result.
2. X: *a num_bootstrapSample by n matrix* which stores the the X values that was sampled using residual resampling bootstrap analyses (In terms of model fitting this is the parameter combinations with variabilities that represent **parameter estimation uncertainties**.).
3. Y: *a num_bootstrapSample by m matrix* which stores the nonlinearFunction evaluated at the corresponding bootstrap analyses results in matrix bootstrapX above. In the context of model fitting each row corresponds to **the model simulations**.
4. runSetting: identical to what is given as CGNM_result but in addition including num_bootstrapSample and indicesToUseAsInitialIterates.

Examples

```
##lip-flop kinetics (an example known to have two distinct solutions)

model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
```

```
nonlinearFunction=model_analytic_function,
targetVector = observation, num_iteration = 10, num_minimizersToFind = 100,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
lowerBound=rep(0,3), ParameterNames=c("Ka","V1","CL_2"), saveLog = FALSE)

CGNM_bootstrap=Cluster_Gauss_Newton_Bootstrap_method(CGNM_result,
     nonlinearFunction=model_analytic_function, num_bootstrapSample=100)

plot_paraDistribution_byHistogram(CGNM_bootstrap)
```

`Cluster_Gauss_Newton_method`

Cluster_Gauss_Newton_method

Description

Find multiple minimisers of the nonlinear least squares problem.

$$argmin_x||f(x)-y*||$$

where

1. f: nonlinear function (e.g., mathematical model)
2. y*: target vector (e.g., observed data to fit the mathematical model)
3. x: variable of the nonlinear function that we aim to find the values that minimize (minimizers) the differences between the nonlinear function and target vector (e.g., model parameter)

Parameter estimation problems of mathematical models can often be formulated as nonlinear least squares problems. In this context f can be thought at a model, x is the parameter, and y* is the observation. CGNM iteratively estimates the minimizer of the nonlinear least squares problem from various initial estimates hence finds multiple minimizers. Full detail of the algorithm and comparison with conventional method is available in the following publication, also please cite this publication when this algorithm is used in your research: Aoki et al. (2020) <doi.org/10.1007/s11081-020-09571-2>. Cluster Gauss–Newton method. Optimization and Engineering, 1-31. As illustrated in this paper, CGNM is faster and more robust compared to repeatedly applying the conventional optimization/nonlinear least squares algorithm from various initial estimates. In addition, CGNM can realize this speed assuming the nonlinear function to be a black-box function (e.g. does not use things like adjoint equation of a system of ODE as the function does not have to be based on a system of ODEs.).

Usage

```
Cluster_Gauss_Newton_method(
  nonlinearFunction,
  targetVector,
  initial_lowerRange,
  initial_upperRange,
```

```
  lowerBound = NA,
  upperBound = NA,
  ParameterNames = NA,
  stayIn_initialRange = FALSE,
  num_minimizersToFind = 250,
  num_iteration = 25,
  saveLog = TRUE,
  runName = "",
  textMemo = "",
  algorithmParameter_initialLambda = 1,
  algorithmParameter_gamma = 2,
  algorithmVersion = 3,
  initialIterateMatrix = NA,
  targetMatrix = NA,
  weightMatrix = NA,
  keepInitialDistribution = NA,
  MO_weights = NA,
  MO_values = NA,
  ...
)
```

Arguments

`nonlinearFunction`	(required input) *A function with input of a vector x of real number of length n and output a vector y of real number of length m.* In the context of model fitting the nonlinearFunction is **the model**. Given the CGNM does not assume the uniqueness of the minimizer, m can be less than n. Also CGNM does not assume any particular form of the nonlinear function and also does not require the function to be continuously differentiable (see Appendix D of our publication for an example when this function is discontinuous). Also this function can be matrix to matrix equation. This can be used for parallerization, see vignettes for examples.
`targetVector`	(required input) *A vector of real number of length m* where we minimize the Euclidean distance between the nonlinearFuncition and targetVector. In the context of curve fitting targetVector can be though as **the observational data**.
`initial_lowerRange`	(required input) *A vector of real number of length n* where each element represents **the lower range of the initial iterate**. Similarly to regular Gauss-Newton method, CGNM iteratively reduce the residual to find minimizers. Essential differences is that CGNM start from the initial RANGE and not an initial point.
`initial_upperRange`	(required input) *A vector of real number of length n* where each element represents **the upper range of the initial iterate**.
`lowerBound`	(default: NA) *A vector of real number or NA of length n* where each element represents **the lower bound of the parameter search**. If no lower bound set that element NA. Note that CGNM is an unconstraint optimization method so the final minimizer can be anywhere. In the parameter estimation problem, there

	often is a constraints to the parameters (e.g., parameters cannot be negative). So when the upper or lower bound is set using this option, parameter transformation is conducted internally (e.g., if either the upper or lower bound is given parameters are log transformed, if the upper and lower bounds are given logit transform is used.)
`upperBound`	(default: NA) *A vector of real number or NA of length n* where each element represents **the upper bound of the parameter search**. If no upper bound set that element NA.
`ParameterNames`	(default: NA) *A vector of string* of length n User can specify names of the parameters that will be used for the plots.
`stayIn_initialRange`	(default: FALSE) *TRUE or FALSE* if set TRUE, the parameter search will conducted strictly within the range specified by initial_lowerRange and initial_upperRange.
`num_minimizersToFind`	(default: 250) *A positive integer* defining number of approximate minimizers CGNM will find. We usually **use 250 when testing the model and 1000 for the final analysis**. The computational cost increase proportionally to this number; however, larger number algorithm becomes more stable and increase the chance of finding more better minimizers. See Appendix C of our paper for detail.
`num_iteration`	(default: 25) *A positive integer* defining maximum number of iterations. We usually **set 25 while model building and 100 for final analysis**. Given each point terminates the computation when the convergence criterion is met the computation cost does not grow proportionally to the number of iterations (hence safe to increase this without significant increase in the computational cost).
`saveLog`	(default: TRUE) *TRUE or FALSE* indicating either or not to save computation result from each iteration in CGNM_log folder. It requires disk write access right in the current working directory. **Recommended to set TRUE if the computation is expected to take long time** as user can retrieve intrim computation result even if the computation is terminated prematurely (or even during the computation).
`runName`	(default: "") *string* that user can ue to identify the CGNM runs. The run history will be saved in the folder name CGNM_log_<runName>. If this is set to "TIME" then runName is automatically set by the run start time.
`textMemo`	(default: "") *string* that user can write an arbitrary text (without influencing computation). This text is stored with the computation result so that can be used for example to describe model so that the user can recognize the computation result.
`algorithmParameter_initialLambda`	(default: 1) *A positive number* for initial value for the regularization coefficient lambda see Appendix B of of our paper for detail.
`algorithmParameter_gamma`	(default: 2) *A positive number* a positive scalar value for adjusting the strength of the weighting for the linear approximation see Appendix A of our paper for detail.

`algorithmVersion`	(default: 3.0) *A positive number* user can choose different version of CGNM algorithm currently 1.0 and 3.0 are available. If number chosen other than 1.0 or 3.0 it will choose 1.0.
`initialIterateMatrix`	(default: NA) *A matrix* with dimension num_minimizersToFind x n. User can provide initial iterate as a matrix This input is used when the user wishes not to generate initial iterate randomly from the initial range. The user is responsible for ensuring all function evaluation at each initial iterate does not produce NaN.
`targetMatrix`	(default: NA) *A matrix* with dimension num_minimizersToFind x m User can define multiple target vectors in the matrix form. This input is mainly used when running bootstrap method and not intended to be used for other purposes.
`weightMatrix`	(default: NA) *A matrix* with dimension num_minimizersToFind x m User can define multiple weight vectors in the matrix form to weight the observations. This input is mainly used when running case sampling bootstrap method and not intended to be used for other purposes.
`keepInitialDistribution`	(default: NA) *A vector of TRUE or FALSE* of length n User can specify if the initial distribution of one of the input variable (e.g. parameter) to be kept as the initial iterate throughout CGNM iterations.
`MO_weights`	(default: NA) *A numeric vector* where the weights for the middle out methods are specified. The length of the vector should be the same as the number of parameters. MO can be used to incoperate prior knowledge of the parameter to be estimated, weight indicate an arbitrary confidence for the prior information. (MO method is still under methodological development.)
`MO_values`	(default: NA) *A numeric vector* where the values for the middle out methods are specified. The length of the vector should be the same as the number of parameters. MO can be used to incoperate prior knowledge of the parameter to be estimated. (MO method is still under methodological development.)
`...`	Further arguments to be supplied to nonlinearFunction

Value

list of a matrix X, Y,residual_history and initialX, as well as a list runSetting

1. X: *a num_minimizersToFind by n matrix* which stores the approximate minimizers of the nonlinear least squares in each row. In the context of model fitting they are **the estimated parameter sets**.
2. Y: *a num_minimizersToFind by m matrix* which stores the nonlinearFunction evaluated at the corresponding approximate minimizers in matrix X above. In the context of model fitting each row corresponds to **the model simulations**.
3. residual_history: *a num_iteration by num_minimizersToFind matrix* storing sum of squares residual for all iterations.
4. initialX: *a num_minimizersToFind by n matrix* which stores the set of initial iterates.
5. runSetting: a list containing all the input variables to Cluster_Gauss_Newton_method (i.e., nonlinearFunction, targetVector, initial_lowerRange, initial_upperRange ,algorithmParameter_initialLambda, algorithmParameter_gamma, num_minimizersToFind, num_iteration, saveLog, runName, textMemo).

Examples

```
##lip-flop kinetics (an example known to have two distinct solutions)

model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation, num_iteration = 10, num_minimizersToFind = 100,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
saveLog = FALSE)

acceptedApproximateMinimizers(CGNM_result)

## Not run:
library(RxODE)

model_text="
d/dt(X_1)=-ka*X_1
d/dt(C_2)=(ka*X_1-CL_2*C_2)/V1"

model=RxODE(model_text)
#define nonlinearFunction
model_function=function(x){

observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)

theta <- c(ka=x[1],V1=x[2],CL_2=x[3])
ev <- eventTable()
ev$add.dosing(dose = 1000, start.time =0)
ev$add.sampling(observation_time)
odeSol=model$solve(theta, ev)
log10(odeSol[,"C_2"])

}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))
```

```
CGNM_result=Cluster_Gauss_Newton_method(nonlinearFunction=model_function,
targetVector = observation, saveLog = FALSE,
initial_lowerRange = c(0.1,0.1,0.1),initial_upperRange =  c(10,10,10))
## End(Not run)
```

`col_quantile` *col_quantile*

Description

Obtain columb wise quantile

Usage

```
col_quantile(data_in, prob)
```

Arguments

`data_in`	(required input) *a matrix or a data.grame* where the column-wise quantile wishes to be determined.
`prob`	(required input) *a number* quantile expressed as in the probability.

Value

a vector of number dim(data_in)[2] containing: quantile of the each column where the probability is specified as "prob"

Examples

```
A=matrix(seq(1,100),nrow = 25)
col_quantile(A, 0.5)
```

`plot_2DprofileLikelihood` *plot_2DprofileLikelihood*

Description

Make likelihood related values v.s. parameterValues plot using the function evaluations used during CGNM computation. Note plot_SSRsurface can only be used when log is saved by setting saveLog=TRUE option when running Cluster_Gauss_Newton_method().

Usage

```
plot_2DprofileLikelihood(
  logLocation,
  index_x = NA,
  index_y = NA,
  plotType = 2,
  plotMax = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA,
  numBins = NA,
  showInitialRange = TRUE,
  alpha = 0.25,
  Likelihood_function = Residual_function_def
)
```

Arguments

`logLocation`	(required input) *A string or a list of strings* of folder directory where CGNM computation log files exist.
`index_x`	(default: NA) *A vector of strings or numbers* List parameter names or indices used for the surface plot. (if NA all parameters are used)
`index_y`	(default: NA) *A vector of strings or numbers* List parameter names or indices used for the surface plot. (if NA all parameters are used)
`plotType`	(default: 2) *A number 0,1,2,3, or 4* 0: number of model evaluations done, 1: (1-alpha) where alpha is the significance level, this plot is recommended for the ease of visualization as it ranges from 0 to 1. 2: -2log likelihood. 3: SSR. 4: all points within 1-alpha confidence region
`plotMax`	(default: NA) *A number* the maximum value that will be plotted on surface plot. (If NA all values are included in the plot, note SSR or likelihood can range many orders of magnitudes fo may want to restrict when plotting them)
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax
`numBins`	(default: NA) *A positive integer* 2D profile likelihood surface is plotted by finding the minimum SSR given two of the parameters are fixed and then repeat this for various values. numBins specifies the number of different parameter values to fix for each parameter. (if set NA the number of bins are set as num_minimizersToFind/10)
`showInitialRange`	(default: TRUE) *TRUE or FALSE* if TRUE then the initial range appears in the plot.
`alpha`	(default: 0.25) *a number between 0 and 1* level of significance (all the points outside of this significance level will not be plotted when plot tyoe 1,2 or 4 are chosen).

`Likelihood_function`

(default: Residual_function_def) *a function* that takes CGNM_result and initial then to calculate a quantity to be sketched in logscale e.g. SSR) this was implemented to conduct pos hoc drawing of the profile likelihood by providing the new definition of likelihood after all CGNM calculations are done.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=10^x[1]
 V1=10^x[2]
 CL_2=10^x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(-1,-1,-1), initial_upperRange =  c(1,1,1),
num_iter = 10, num_minimizersToFind = 500, saveLog=TRUE)

## the minimum example
plot_2DprofileLikelihood("CGNM_log")

## we can draw profilelikelihood also including bootstrap result
CGNM_result=Cluster_Gauss_Newton_Bootstrap_method(CGNM_result,
                      nonlinearFunction = model_analytic_function)

## example with various options
plot_2DprofileLikelihood(c("CGNM_log","CGNM_log_bootstrap"),
 showInitialRange = TRUE,index_x = c("ka","V1"))

## End(Not run)
```

`plot_goodnessOfFit` *plot_goodnessOfFit*

Description

Make goodness of fit plots to assess the model-fit and bias in residual distribution. The linear model is fit to the residual and plotted using geom_smooth(method=lm) in ggplot.

Explanation of the terminologies in terms of PBPK model fitting to the time-course drug concentration measurements:
"independent variable" is time
"dependent variable" is the concentration.
"Residual" is the difference between the measured concentration and the model simulation with the parameter fond by the CGNM.
"m" is number of observations

Usage

```
plot_goodnessOfFit(
  CGNM_result,
  plotType = 1,
  plotRank = c(1),
  independentVariableVector = NA,
  dependentVariableTypeVector = NA,
  absResidual = FALSE
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`plotType`	(default: 1) *1,2 or 3* specify the kind of goodness of fit plot to create
`plotRank`	(default: c(1)) *a vector of integers* Specify which rank of the parameter to use for the goodness of fit plots. (e.g., if one wishes to use rank 1 to 100 then set it to be seq(1,100), or if one wish to use 88th rank parameters then set this as 88.)
`independentVariableVector`	(default: NA) *a vector of numerics of length m* set independent variables that target values are associated with (e.g., time of the drug concentration measurement one is fitting PBPK model to) (when this variable is set to NA, seq(1,m) will be used as independent variable when appropriate).
`dependentVariableTypeVector`	(default: NA) *a vector of text of length m* when this variable is set (i.e., not NA) then the goodness of fit analyses is done

	for each variable type. For example, if we are fitting the PBPK model to data with multiple dose arms, one can see the goodness of fit for each dose arm by specifying which dose group the observations are from.
`absResidual`	(default: FALSE) *TRUE or FALSE* If TRUE plot absolute values of the residual.

Value

A ggplot object of the goodness of fit plot.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=10^x[1]
 V1=10^x[2]
 CL_2=10^x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = rep(0.01,3), initial_upperRange =  rep(100,3),
lowerBound=rep(0,3), ParameterNames = c("Ka","V1","CL"),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

plot_goodnessOfFit(CGNM_result)
plot_goodnessOfFit(CGNM_result,
     independentVariableVector=c(0.1,0.2,0.4,0.6,1,2,3,6,12))
```

`plot_paraDistribution_byHistogram`
plot_paraDistribution_byHistogram

Description

Make histograms to visualize the initial distribution and distribition of the accepted approximate minimizers found by the CGNM.

Usage

```
plot_paraDistribution_byHistogram(
  CGNM_result,
  indicesToInclude = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA,
  bins = 30
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`indicesToInclude`	(default: NA) *A vector of integers* indices to include in the plot (if NA, use indices chosen by the acceptedIndices() function with default setting).
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax
`bins`	(default: 30) *A natural number* Number of bins used for plotting histogram.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))
```

```
CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = rep(0.01,3), initial_upperRange =  rep(100,3),
lowerBound=rep(0,3), ParameterNames = c("Ka","V1","CL"),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

plot_paraDistribution_byHistogram(CGNM_result)
plot_paraDistribution_byHistogram(CGNM_result,
     ReparameterizationDef=c("log10(Ka)","log10(V1)","log10(CL)"))
```

`plot_paraDistribution_byViolinPlots` *plot_paraDistribution_byViolinPlots*

Description

Make violin plot to compare the initial distribution and distribition of the accepted approximate minimizers found by the CGNM. Bars in the violin plots indicates the interquartile range. The solid line connects the interquartile ranges of the initial distribution and the distribution of the accepted approximate minimizer at the final iterate. The blacklines connets the minimums and maximums of the initial distribution and the distribution of the accepted approximate minimizer at the final iterate. The black dots indicate the median.

Usage

```
plot_paraDistribution_byViolinPlots(
  CGNM_result,
  indicesToInclude = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`indicesToInclude`	(default: NA) *A vector of integers* indices to include in the plot (if NA, use indices chosen by the acceptedIndices() function with default setting).
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))



CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = rep(0.01,3), initial_upperRange =  rep(100,3),
lowerBound=rep(0,3), ParameterNames = c("Ka","V1","CL"),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

plot_paraDistribution_byViolinPlots(CGNM_result)
plot_paraDistribution_byViolinPlots(CGNM_result,
     ReparameterizationDef=c("log10(Ka)","log10(V1)","log10(CL)"))
```

`plot_parameterValue_scatterPlots`

plot_parameterValue_scatterPlots

Description

Make scatter plots of the accepted approximate minimizers found by the CGNM. Bars in the violin plots indicates the interquartile range.

Usage

```
plot_parameterValue_scatterPlots(CGNM_result, indicesToInclude = NA)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`indicesToInclude`	(default: NA) *A vector of integers* indices to include in the plot (if NA, use indices chosen by the acceptedIndices() function with default setting).

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

plot_parameterValue_scatterPlots(CGNM_result)
```

`plot_profileLikelihood` *plot_profileLikelihood*

Description

Draw profile likelihood surface using the function evaluations conducted during CGNM computation. Note plot_SSRsurface can only be used when log is saved by setting saveLog=TRUE option when running Cluster_Gauss_Newton_method(). The grey horizontal line is the threshold for 95

Usage

```
plot_profileLikelihood(
  logLocation,
  alpha = 0.25,
  numBins = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA,
  showInitialRange = TRUE,
  Likelihood_function = Residual_function_def
)
```

Arguments

`logLocation`	(required input) *A string* of folder directory where CGNM computation log files exist.
`alpha`	(default: 0.25) *a number between 0 and 1* level of significance (used to draw horizontal line on the profile likelihood).
`numBins`	(default: NA) *A positive integer* SSR surface is plotted by finding the minimum SSR given one of the parameters is fixed and then repeat this for various values. numBins specifies the number of different parameter values to fix for each parameter. (if set NA the number of bins are set as num_minimizersToFind/10)
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax
`showInitialRange`	(default: TRUE) *TRUE or FALSE* if TRUE then the initial range appears in the plot.
`Likelihood_function`	(default: Residual_function_def) *a function* that takes CGNM_result and initial then to calculate a quantity to be sketched in logscale e.g. SSR) this was implemented to conduct pos hoc drawing of the profile likelihood by providing the new definition of likelihood after all CGNM calculations are done.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1
```

```
 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog=TRUE)

plot_profileLikelihood("CGNM_log")

## End(Not run)
```

`plot_Rank_SSR` *plot_Rank_SSR*

Description

Make SSR v.s. rank plot. This plot is often used to visualize the maximum accepted SSR.

Usage

```
plot_Rank_SSR(CGNM_result, indicesToInclude = NA)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`indicesToInclude`	(default: NA) *A vector of integers* indices to include in the plot (if NA, use indices chosen by the acceptedIndices() function with default setting).

Value

A ggplot object of SSR v.s. rank.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

plot_Rank_SSR(CGNM_result)
```

`plot_simulationMatrixWithCI` *plot_simulationMatrixWithCI*

Description

Plot simulation that are provided to plot confidence interval (or more like a confidence region).

Usage

```
plot_simulationMatrixWithCI(
  simulationMatrix,
  independentVariableVector = NA,
  dependentVariableTypeVector = NA,
  confidenceLevels = c(0.25, 0.75),
  observationVector = NA,
  observationIndpendentVariableVector = NA,
  observationDependentVariableTypeVector = NA
)
```

Arguments

`simulationMatrix`
: (required input) *A matrix of numbers* where each row contains the simulated values that will be plotted.

`independentVariableVector`
: (default: NA) *A vector of numbers* that represents the independent variables of each points of the simulation (e.g., observation time) where used for the values of x-axis when plotting. If set at NA then sequence of 1,2,3,... will be used.

`dependentVariableTypeVector`
: (default: NA) *A vector of strings* specify the kind of variable the simulation values are. (i.e., if it simulate both PK and PD then indicate which simulation value is PK and which is PD).

`confidenceLevels`
: (default: c(25,75)) *A vector of two numbers between 0 and 1* set the confidence interval that will be used for the plot. Default is inter-quartile range.

`observationVector`
: (default: NA) *A vector of numbers* used when wishing to overlay the plot of observations to the simulation.

`observationIndpendentVariableVector`
: (default: NA) *A vector of numbers* used when wishing to overlay the plot of observations to the simulation.

`observationDependentVariableTypeVector`
: (default: NA) *A vector of numbers* used when wishing to overlay the plot of observations to the simulation.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 (Cp)
}

observation=(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))
```

```
CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation, num_iteration = 10, num_minimizersToFind = 100,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
lowerBound=rep(0,3), ParameterNames=c("Ka","V1","CL_2"), saveLog = FALSE)

CGNM_bootstrap=Cluster_Gauss_Newton_Bootstrap_method(CGNM_result,
     nonlinearFunction=model_analytic_function, num_bootstrapSample=100)


plot_simulationMatrixWithCI(CGNM_result$bootstrapY,
independentVariableVector=observation_time, observationVector=observation)

## End(Not run)
```

`plot_simulationWithCI` *plot_simulationWithCI*

Description

Plot model simulation where the various parameter combinations are provided and conduct simulations and then the confidence interval (or more like a confidence region) is plotted.

Usage

```
plot_simulationWithCI(
  simulationFunction,
  parameter_matrix,
  independentVariableVector = NA,
  dependentVariableTypeVector = NA,
  confidenceLevels = c(0.25, 0.75),
  observationVector = NA,
  observationIndpendentVariableVector = NA,
  observationDependentVariableTypeVector = NA
)
```

Arguments

`simulationFunction`	(required input) *A function* that maps the parameter vector to the simulation.
`parameter_matrix`	(required input) *A matrix of numbers* where each row contains the parameter combination that will be used for the simulations.
`independentVariableVector`	(default: NA) *A vector of numbers* that represents the independent variables of each points of the simulation (e.g., observation time) where used for the values of x-axis when plotting. If set at NA then sequence of 1,2,3,... will be used.

`dependentVariableTypeVector`
: (default: NA) *A vector of strings* specify the kind of variable the simulationFunction simulate out. (i.e., if it simulate both PK and PD then indicate which simulation output is PK and which is PD).

`confidenceLevels`
: (default: c(25,75)) *A vector of two numbers between 0 and 1* set the confidence interval that will be used for the plot. Default is inter-quartile range.

`observationVector`
: (default: NA) *A vector of numbers* used when wishing to overlay the plot of observations to the simulation.

`observationIndpendentVariableVector`
: (default: NA) *A vector of numbers* used when wishing to overlay the plot of observations to the simulation.

`observationDependentVariableTypeVector`
: (default: NA) *A vector of numbers* used when wishing to overlay the plot of observations to the simulation.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 (Cp)
}

observation=(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation, num_iteration = 10, num_minimizersToFind = 100,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
lowerBound=rep(0,3), ParameterNames=c("Ka","V1","CL_2"), saveLog = FALSE)

CGNM_bootstrap=Cluster_Gauss_Newton_Bootstrap_method(CGNM_result,
     nonlinearFunction=model_analytic_function, num_bootstrapSample=100)
```

```
plot_simulationWithCI(model_analytic_function, as.matrix(CGNM_result$bootstrapTheta),
independentVariableVector=observation_time, observationVector=observation)

## End(Not run)
```

`plot_SSRsurface` *plot_SSRsurface*

Description

Make minimum SSR v.s. parameterValue plot using the function evaluations used during CGNM computation. Note plot_SSRsurface can only be used when log is saved by setting saveLog=TRUE option when running Cluster_Gauss_Newton_method().

Usage

```
plot_SSRsurface(
  logLocation,
  alpha = 0.25,
  profile_likelihood = FALSE,
  numBins = NA,
  maxSSR = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA,
  showInitialRange = FALSE,
  Residual_function = Residual_function_def
)
```

Arguments

`logLocation`	(required input) *A string or a list of strings* of folder directory where CGNM computation log files exist.
`alpha`	(default: 0.25) *a number between 0 and 1* level of significance (used to draw horizontal line on the profile likelihood).
`profile_likelihood`	(default: FALSE) *TRUE or FALSE* If set TRUE plot profile likelihood (assuming normal distribution of residual) instead of SSR surface.
`numBins`	(default: NA) *A positive integer* SSR surface is plotted by finding the minimum SSR given one of the parameters is fixed and then repeat this for various values. numBins specifies the number of different parameter values to fix for each parameter. (if set NA the number of bins are set as num_minimizersToFind/10)
`maxSSR`	(default: NA) *A positive number* the maximum SSR that will be plotted on SSR surface plot. This option is used to zoom into the SSR surface near the minimum SSR.
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)

`ReparameterizationDef`
: (default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax

`showInitialRange`
: (default: FALSE) *TRUE or FALSE* if TRUE then the initial range appears in the plot.

`Residual_function`
: (default: Residual_function_def) *a function* that takes CGNM_result and initial then to calculate a quantity to be sketched in logscale e.g. SSR) this was implemented to conduct pos hoc drawing of the profile likelihood by providing the new definition of likelihood after all CGNM calculations are done.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog=TRUE)

plot_SSRsurface("CGNM_log") + scale_y_continuous(trans='log10')

## End(Not run)
```

`plot_SSR_parameterValue`

plot_SSR_parameterValue

Description

Make SSR v.s. parameterValue plot of the accepted approximate minimizers found by the CGNM. Bars in the violin plots indicates the interquartile range.

Usage

```
plot_SSR_parameterValue(
  CGNM_result,
  indicesToInclude = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA,
  showInitialRange = TRUE
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`indicesToInclude`	(default: NA) *A vector of integers* indices to include in the plot (if NA, use indices chosen by the acceptedIndices() function with default setting).
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax
`showInitialRange`	(default: TRUE) *TRUE or FALSE* if TRUE then the initial range appears in the plot.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1
```

```
 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

plot_SSR_parameterValue(CGNM_result)
```

`shinyCGNM` *shinyCGNM*

Description

Start Shiny app that assist the user to make R-script to conduct PBPK model fitting using CGNM.

Usage

```
shinyCGNM()
```

Value

NULL and start graphifcal user interface.

Examples

```
## Not run:
shinyCGNM()

## End(Not run)
```

`suggestInitialLowerRange`

suggestInitialLowerRange

Description

Suggest initial lower range based on the profile likelihood. The user can re-run CGNM with this suggested initial range so that to improve the convergence.

Usage

```
suggestInitialLowerRange(logLocation, alpha = 0.25, numBins = NA)
```

Arguments

`logLocation`	(required input) *A string or a list of strings* of folder directory where CGNM computation log files exist.
`alpha`	(default: 0.25) *a number between 0 and 1* level of significance used to derive the confidence interval.
`numBins`	(default: NA) *A positive integer* SSR surface is plotted by finding the minimum SSR given one of the parameters is fixed and then repeat this for various values. numBins specifies the number of different parameter values to fix for each parameter. (if set NA the number of bins are set as num_minimizersToFind/10)

Value

A numerical vector of suggested initial lower range based on profile likelihood.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))
```

```
CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog=TRUE)

suggestInitialLowerRange("CGNM_log")

## End(Not run)
```

`suggestInitialUpperRange`

suggestInitialUpperRange

Description

Suggest initial upper range based on the profile likelihood. The user can re-run CGNM with this suggested initial range so that to improve the convergence.

Usage

```
suggestInitialUpperRange(logLocation, alpha = 0.25, numBins = NA)
```

Arguments

`logLocation`	(required input) *A string or a list of strings* of folder directory where CGNM computation log files exist.
`alpha`	(default: 0.25) *a number between 0 and 1* level of significance used to derive the confidence interval.
`numBins`	(default: NA) *A positive integer* SSR surface is plotted by finding the minimum SSR given one of the parameters is fixed and then repeat this for various values. numBins specifies the number of different parameter values to fix for each parameter. (if set NA the number of bins are set as num_minimizersToFind/10)

Value

A numerical vector of suggested initial upper range based on profile likelihood.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1
```

```
 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog=TRUE)

suggestInitialLowerRange("CGNM_log")

## End(Not run)
```

`table_parameterSummary` *table_parameterSummary*

Description

Make summary table of the approximate local minimizers found by CGNM. If bootstrap analysis result is available, relative standard error (RSE: standard deviation/mean) will also be included in the table.

Usage

```
table_parameterSummary(
  CGNM_result,
  indicesToInclude = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA
)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`indicesToInclude`	(default: NA) *A vector of integers* indices to include in the plot (if NA, use indices chosen by the acceptedIndices() function with default setting).

`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))



CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = rep(0.01,3), initial_upperRange =  rep(100,3),
lowerBound=rep(0,3), ParameterNames = c("Ka","V1","CL"),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

table_parameterSummary(CGNM_result)
table_parameterSummary(CGNM_result,
     ReparameterizationDef=c("log10(Ka)","log10(V1)","log10(CL)"))
```

`table_profileLikelihoodConfidenceInterval`
table_profileLikelihoodConfidenceInterval

Description

Make table of confidence intervals that are approximated from the profile likelihood. First inspect profile likelihood plot and make sure the plot is smooth and has good enough resolution and the initial range is appropriate. Do not report this table without checking the profile likelihood plot.

Usage

```
table_profileLikelihoodConfidenceInterval(
  logLocation,
  alpha = 0.25,
  numBins = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA,
  pretty = FALSE,
  Likelihood_function = Residual_function_def
)
```

Arguments

`logLocation`	(required input) *A string or a list of strings* of folder directory where CGNM computation log files exist.
`alpha`	(default: 0.25) *a number between 0 and 1* level of significance used to derive the confidence interval.
`numBins`	(default: NA) *A positive integer* SSR surface is plotted by finding the minimum SSR given one of the parameters is fixed and then repeat this for various values. numBins specifies the number of different parameter values to fix for each parameter. (if set NA the number of bins are set as num_minimizersToFind/10)
`ParameterNames`	(default: NA) *A vector of strings* the user can supply so that these names are used when making the plot. (Note if it set as NA or vector of incorrect length then the parameters are named as theta1, theta2, ... or as in ReparameterizationDef)
`ReparameterizationDef`	(default: NA) *A vector of strings* the user can supply definition of reparameterization where each string follows R syntax
`pretty`	(default: FALSE) *TRUE or FALSE* if true then the publication ready table will be an output
`Likelihood_function`	(default: Residual_function_def) *a function* that takes CGNM_result and initial then to calculate a quantity to be sketched in logscale e.g. SSR) this was implemented to conduct pos hoc drawing of the profile likelihood by providing the new definition of likelihood after all CGNM calculations are done.

Value

A ggplot object including the violin plot, interquartile range and median, minimum and maximum.

Examples

```
## Not run:
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog=TRUE)

table_profileLikelihoodConfidenceInterval("CGNM_log")

## End(Not run)
```

`topIndices` *topIndices*

Description

CGNM find multiple sets of minimizers of the nonlinear least squares (nls) problem by solving nls from various initial iterates. Although CGNM is shown to be robust compared to other conventional multi-start algorithms, not all initial iterates minimizes successfully. One can visually inspect rank v.s. SSR plot and manually choose number of best fit acceptable parameters. By using this function "topIndices", we can obtain the indices of the "numTopIndices" best fit parameter combinations.

Usage

```
topIndices(CGNM_result, numTopIndices)
```

Arguments

`CGNM_result`	(required input) *A list* stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
`numTopIndices`	(required input) *An integer* .

Value

A vector of natural number that contains the indices of accepted approximate minimizers found by CGNM.

Examples

```
model_analytic_function=function(x){

 observation_time=c(0.1,0.2,0.4,0.6,1,2,3,6,12)
 Dose=1000
 F=1

 ka=x[1]
 V1=x[2]
 CL_2=x[3]
 t=observation_time

 Cp=ka*F*Dose/(V1*(ka-CL_2/V1))*(exp(-CL_2/V1*t)-exp(-ka*t))

 log10(Cp)
}

observation=log10(c(4.91, 8.65, 12.4, 18.7, 24.3, 24.5, 18.4, 4.66, 0.238))

CGNM_result=Cluster_Gauss_Newton_method(
nonlinearFunction=model_analytic_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1), initial_upperRange =  c(10,10,10),
num_iter = 10, num_minimizersToFind = 100, saveLog = FALSE)

topInd=topIndices(CGNM_result, 10)

## This gives top 10 approximate minimizers
CGNM_result$X[topInd,]
```

Index