

# Package ‘CGNM’

January 10, 2023

**Type** Package

**Title** Cluster Gauss-Newton Method

**Version** 0.6.2

**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 approximate 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](https://doi.org/10.1007/s11081-020-09571-2)>. Cluster Gauss–Newton method. Optimization and Engineering, 1-31.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** stats, ggplot2, MASS

**Suggests** knitr, rmarkdown

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-01-10 06:43:10 UTC

## R topics documented:

acceptedApproximateMinimizers . . . . .	2
acceptedIndices . . . . .	4
acceptedIndices_binary . . . . .	5
acceptedMaxSSR . . . . .	7
bestApproximateMinimizers . . . . .	8
Cluster_Gauss_Newton_Bootstrap_method . . . . .	10
Cluster_Gauss_Newton_method . . . . .	12
plot_2DprofileLikelihood . . . . .	16
plot_goodnessOfFit . . . . .	18

plot_paraDistribution_byHistogram . . . . .	20
plot_paraDistribution_byViolinPlots . . . . .	22
plot_parameterValue_scatterPlots . . . . .	23
plot_profileLikelihood . . . . .	24
plot_Rank_SSR . . . . .	26
plot_SSRsurface . . . . .	27
plot_SSR_parameterValue . . . . .	29
table_parameterSummary . . . . .	30
table_profileLikelihoodConfidenceInterval . . . . .	32
topIndices . . . . .	33

<b>Index</b>	<b>35</b>
--------------	-----------

---

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 chi-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) <i>A natural number</i> 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) <i>TRUE or FALSE</i> 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) <i>1 or 2</i> specify the algorithm used for obtain accepted approximate minimizers. (Algorithm 1 uses elbow method, Algorithm 2 uses Grubbs' Test for Outliers.)
ParameterNames	(default: NA) <i>A vector of strings</i> 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) <i>A vector of strings</i> 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)

acceptedApproximateMinimizers(CGNM_result)

```

---

acceptedIndices	<i>acceptedIndices</i>
-----------------	------------------------

---

### 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 chi-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) <i>A list</i> stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
cutoff_pvalue	(default: 0.05) <i>A number</i> defines the rejection p-value for the first stage of acceptable computational result screening.
numParametersIncluded	(default: NA) <i>A natural number</i> 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) <i>TRUE or FALSE</i> If true then use chi-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) <i>1 or 2</i> 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)

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 chi-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)))[acceptedIndices_binary(CGNM_result)]`)

**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)

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 chi-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 returns numParametersIncluded-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)

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)

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
)

```

## 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 specify 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).

**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"))

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.

$$\operatorname{argmin}_x \|f(x) - y^*\|$$

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 = FALSE,
  runName = "",
  textMemo = "",
  algorithmParameter_initialLambda = 1,
```

```

algorithmParameter_gamma = 2,
algorithmVersion = 3,
initialIterateMatrix = NA,
targetMatrix = NA,
keepInitialDistribution = 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).

targetVector

(required input) A vector of real number of length  $m$  where we minimize the Euclidean distance between the nonlinearFunction and targetVector. In the context of curve fitting targetVector can be thought 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. Note that CGNM is an unconstrained optimization method so the final minimizer can be anywhere (and outside of this specified range). In the parameter estimation problem, there often is a constraints to the parameters (e.g., parameters cannot be negative).. If you wish to constraint the parameter domain do so via parameter transformation (e.g., if parameter needs to be positive do log transform, if there is upper and lower bounds consider using logit transform.)

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.

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 be 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**

	<b>final analysis.</b> 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) <i>A positive integer</i> defining maximum number of iterations. We usually <b>set 25 while model building and 100 for final analysis</b> . 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: FALSE) <i>TRUE or FALSE</i> 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. <b>Recommended to set TRUE if the computation is expected to take long time</b> as user can retrieve intrim computation result even if the computation is terminated prematurely (or even during the computation).
runName	(default: "") <i>string</i> 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: "") <i>string</i> 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) <i>A positive number</i> for initial value for the regularization coefficient lambda see Appendix B of of our paper for detail.
algorithmParameter_gamma	(default: 2) <i>A positive number</i> 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) <i>A positive number</i> 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) <i>A matrix</i> 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) <i>A matrix</i> 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.
keepInitialDistribution	(default: NA) <i>A vector of TRUE or FALSE</i> 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.

### 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))

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,
initial_lowerRange = c(0.1,0.1,0.1),initial_upperRange = c(10,10,10))
## End(Not run)

```

---

```

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 = FALSE,
  alpha = 0.25
)

```



**Arguments**

logLocation	(required input) <i>A string or a list of strings</i> of folder directory where CGNM computation log files exist.
index_x	(default: NA) <i>A vector of strings or numbers</i> List parameter names or indices used for the surface plot. (if NA all parameters are used)
index_y	(default: NA) <i>A vector of strings or numbers</i> List parameter names or indices used for the surface plot. (if NA all parameters are used)
plotType	(default: 2) <i>A number 0,1,2,3, or 4</i> 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) <i>A number</i> 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) <i>A vector of strings</i> 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) <i>A vector of strings</i> the user can supply definition of reparameterization where each string follows R syntax
numBins	(default: NA) <i>A positive integer</i> 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: FALSE) <i>TRUE or FALSE</i> if TRUE then the initial range appears in the plot.
alpha	(default: 0.25) <i>a number between 0 and 1</i> level of significance (all the points outside of this significance level will not be plotted when plot tyoe 1,2 or 4 are chosen).

**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.

Explanation of the terminologies in terms of PBPK model fitting to the time-course drug concentration measurements:

1. "independent variable" is time
2. "dependent variable" is the concentration.
3. "Residual" is the difference between the measured concentration and the model simulation with the parameter found by the CGNM.
4. "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) <i>A list</i> stores the computational result from Cluster_Gauss_Newton_method() function in CGNM package.
plotType	(default: 1) <i>1, 2 or 3</i> specify the kind of goodness of fit plot to create <ol style="list-style-type: none"> <li>1. dependent variable v.s. independent variable with overlay of the target as red dots (e.g., plots of time-course concentration profile with overlay of observed concentration in red dots). When CGNM_result include bootstrap analysis result, then the model simulation with median, 5 percentile and 95 percentile will be plotted.</li> <li>2. residual v.s. dependent variable (used to check to make sure one has chosen the right "shape" of residual distribution, i.e., additive, proportional etc., check to make sure there is no noticeable trend.)</li> <li>3. residual v.s. independent variable (e.g., use to check if the model-fit is equally good throughout different phases of time-course profile.)</li> </ol>
plotRank	(default: c(1)) <i>an integer of a vector of integers</i> 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) <i>a vector of numerics of length m</i> 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) <i>a vector of text of length m</i> 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) <i>TRUE or FALSE</i> 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)

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 distribution 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)

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 distribution 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)

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)

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 = FALSE
)
```

**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: FALSE) *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**

```
## 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)

plot_Rank_SSR(CGNM_result)

```

---

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
)

```

## Arguments

logLocation	(required input) <i>A string or a list of strings</i> of folder directory where CGNM computation log files exist.
alpha	(default: 0.25) <i>a number between 0 and 1</i> level of significance (used to draw horizontal line on the profile likelihood).

profile_likelihood	(default: FALSE) <i>TRUE or FALSE</i> If set TRUE plot profile likelihood (assuming normal distribution of residual) instead of SSR surface.
numBins	(default: NA) <i>A positive integer</i> 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) <i>A positive number</i> 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) <i>A vector of strings</i> 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) <i>A vector of strings</i> the user can supply definition of reparameterization where each string follows R syntax
showInitialRange	(default: FALSE) <i>TRUE or FALSE</i> 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

```
## 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)

plot_SSR_parameterValue(CGNM_result)

```

---

```
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)

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

### Usage

```
table_profileLikelihoodConfidenceInterval(
  logLocation,
  alpha = 0.25,
  numBins = NA,
  ParameterNames = NA,
  ReparameterizationDef = NA
)
```

### Arguments

logLocation	(required input) <i>A string or a list of strings</i> of folder directory where CGNM computation log files exist.
alpha	(default: 0.25) <i>a number between 0 and 1</i> level of significance used to derive the confidence interval.
numBins	(default: NA) <i>A positive integer</i> 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) <i>A vector of strings</i> 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) <i>A vector of strings</i> 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

```
## 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)  
  
topInd=topIndices(CGNM_result, 10)  
  
## This gives top 10 approximate minimizers  
CGNM_result$X[topInd,]
```

# Index

acceptedApproximateMinimizers, [2](#)  
acceptedIndices, [4](#)  
acceptedIndices\_binary, [5](#)  
acceptedMaxSSR, [7](#)  
  
bestApproximateMinimizers, [8](#)  
  
Cluster\_Gauss\_Newton\_Bootstrap\_method,  
[10](#)  
Cluster\_Gauss\_Newton\_method, [12](#)  
  
plot\_2DprofileLikelihood, [16](#)  
plot\_goodnessOfFit, [18](#)  
plot\_paraDistribution\_byHistogram, [20](#)  
plot\_paraDistribution\_byViolinPlots,  
[22](#)  
plot\_parameterValue\_scatterPlots, [23](#)  
plot\_profileLikelihood, [24](#)  
plot\_Rank\_SSR, [26](#)  
plot\_SSR\_parameterValue, [29](#)  
plot\_SSRsurface, [27](#)  
  
table\_parameterSummary, [30](#)  
table\_profileLikelihoodConfidenceInterval,  
[32](#)  
topIndices, [33](#)