Package 'CGNM'

March 14, 2022

Type Package

Title Cluster Gauss-Newton Method

Version 0.3.1

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>. Cluster Gauss–Newton method. Optimization and Engineering, 1-31.

License MIT + file LICENSE

Encoding UTF-8

Imports stats, ggplot2, MASS

Suggests knitr, rmarkdown, RxODE

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-14 19:30:06 UTC

R topics documented:

`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
)
```

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).

Value

A matrix 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)
```

`acceptedIndecies` *acceptedIndecies*

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

```
acceptedIndecies(
  CGNM_result,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
```

```
  useAcceptedApproximateMinimizers = TRUE
)
```

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).

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)
```

```
acceptedIndecies(CGNM_result)
```

`acceptedIndecies_binary` *acceptedIndecies_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 acceptedIndecies(CGNM_result) is equal to seq(1,length(acceptedIndecies_binary(CGNM_result)))[accep

Usage

```
acceptedIndecies_binary(
  CGNM_result,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  useAcceptedApproximateMinimizers = TRUE
)
```

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).

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)

acceptedIndecies_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
)
```

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).

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)
```

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) .

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,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  useAcceptedApproximateMinimizers = TRUE
)
```

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.

`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).

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_Bootstrap_method(nonlinearFunction=model_function,
targetVector = observation,
initial_lowerRange = c(0.1,0.1,0.1),initial_upperRange =  c(10,10,10))
## End(Not run)
```

`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,
  num_minimizersToFind = 250,
  num_iteration = 25,
  saveLog = FALSE,
  runName = "",
  textMemo = "",
  algorithmParameter_initialLambda = 1,
  algorithmParameter_gamma = 2,
  algorithmVersion = 3,
  initialIterateMatrix = NA,
  targetMatrix = 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 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. Note that CGNM is an unconstraint 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**.

`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: FALSE) *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.

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_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,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  ParameterNames = NA,
  useAcceptedApproximateMinimizers = TRUE
)
```

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.
ParameterNames	(default: NA) *A vector of string* 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 x_1, x_2, ...)
useAcceptedApproximateMinimizers	(default: TRUE) *TRUE or FALSE* If true then use chai-square and elbow method to choose maximum accepted SSR. If false use parameters upto numParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then use all the parameters found by the CGNM).

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_paraDistribution_byViolinPlots(CGNM_result)
```

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,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  ParameterNames = NA,
  useAcceptedApproximateMinimizers = TRUE
)
```

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.
ParameterNames	(default: NA) *A vector of string* 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 x_1, x_2, ...)
useAcceptedApproximateMinimizers	(default: TRUE) *TRUE or FALSE* If true then use chai-square and elbow method to choose maximum accepted SSR. If false use parameters upto numParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then use all the parameters found by the CGNM).

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_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,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  useAcceptedApproximateMinimizers = TRUE
)
```

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 use parameters upto numParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then use all the parameters found by the CGNM).

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_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,
  cutoff_pvalue = 0.05,
  numParametersIncluded = NA,
  ParameterNames = NA,
  showInitialRange = TRUE,
  useAcceptedApproximateMinimizers = TRUE
)
```

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.
`ParameterNames`	(default: NA) *A vector of string* 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 x_1, x_2, ...)
`showInitialRange`	(default: TRUE) *TRUE or FALSE* if TRUE then the initial range appears in the plot.
`useAcceptedApproximateMinimizers`	(default: TRUE) *TRUE or FALSE* If true then use chai-square and elbow method to choose maximum accepted SSR. If false use parameters upto numParametersIncluded-th smallest SSR (or if numParametersIncluded=NA then use all the parameters found by the CGNM).

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)
```

Index