Package KinGUI2

March 14, 2012

`autolayout` *Set the layout when making multiple plots on the same page.*

Description

Utility function to set the layout when making multiple plots on the same page.

Usage

```
autolayout(n)
```

Arguments

`n`	number of plots to make

Value

number of rows and columns in the layout

Author(s)

Zhenglei Gao

`chi2err` *Calculate model error at which chi-square test is passed.*

Description

This function calculates the smallest relative error resulting in passing the chi-squared test as defined in the FOCUS kinetics report from 2006.

Usage

```
chi2err(errdata, n.parms, errobserved, alpha = 0.05)
```

Arguments

errdata	A data frame with mean observed values in column named 'value_mean' and predicted values in column 'value_pred'.
n.parms	The number of optimized parameters to be taken into account for the data series.
errobserved	A data frame with the true observed values instead of the mean values for repeated measurements.
alpha	The confidence level chosen for the chi-squared test.

Details

This function is used internally by mkinfit.full, IRLSkinfit.full, and mcmckinfit.full.

Value

A list of component to assess the model goodness-of-fit including:

err.min	Chi2 error level
n.optim	Number of parameters
df	Degree of freedom
RMSE	The root mean squared error.
EF	Model efficiency
R2	Coefficient of determination

Author(s)

Zhenglei Gao

---

completeCompound — *Auxiliary function including error checks for* mkinmod.full

Description

Auxiliary function including error checks for mkinmod.full

Usage

```
completeCompound(compound = list(type = "SFO", to = "M1"),
  varname = NULL, first = FALSE,
  inpartri = c("default", "water-sediment", "advanced"),
  outpartri = c("default", "water-sediment", "advanced"),
  data = NULL, weight = NULL, update = NULL, ...)
```

Arguments

`compound`	A list of properties for a single compound.
`varname`	The compound name.
`first`	hether this compound is the parent compound
`inpartri`	Input parameterization.
`outpartri`	Output parameterization.
`data`	If not NULL, The residue data frame.
`weight`	If weight is NULL, check weight component.
`update`	If not NULL, replace the components in compound with the ones in the update.
`...`	Other optional arguments. Not used.

Value

A list of components to be used in `mkinmod.full`. For the differential functions related to other compounds, they are derived in the `mkinmod.full` function.

Author(s)

Zhenglei Gao

EF — *Calculating the Model Efficiency*

Description

Calculating the Model Efficiency according to Focus Kinetics Guidelines.

Usage

```
EF(o, c)
```

Arguments

`o`	Observed values.
`c`	Predicted values using model.

Details

For EF>0, the values gives an indication of fraction of the total variance of the data that can be explained by the model.

Value

Model efficiency(EF) ranges from negative infinity to 1.

Author(s)

Zhenglei Gao

IRLSkinfit.full *Fit a kinetic model using the IRLS algorithm.*

Description

Fit a kinetic model using the IRLS algorithm.

Usage

```
IRLSkinfit.full(mkinmodini, eigen = FALSE, plot = FALSE,
  plottitle = "", quiet = FALSE, err = NULL,
  weight = "none", scaleVar = FALSE,
  ctr = kingui.control(), irls.control = list(),
  update = NULL, useHsolnp = FALSE, ...)
```

Arguments

`mkinmodini` A list of class `mkinmod.full`, containing the kinetic model to be fitted to the data, and the initial parameter values, the observed data.

`eigen` If TRUE, the solution of the system of differential equation should be based on the spectral decomposition of the coefficient matrix in cases that this is possible.

`plot` If TRUE,the observed values and the numerical solutions should be plotted at each stage of the optimisation.

`plottitle` The title of the plot for visualizing the optimization process.

`quiet` If TRUE, suppress printing out the current model cost after each(>1) improvement.

`err` See argumetns of `mkinfit.full`

`weight` See argumetns of `mkinfit.full`

`scaleVar` See argumetns of `mkinfit.full`

`ctr` A list of control values for the estimation algorithm to replace the default values including maximum iterations and absolute error tolerance. Defaults to the output of `kingui.control`.

`irls.control` A list of control values for the estimation algorithm to replace the default values including the maximum number of iterations for the outer iteration and the error tolerance level for the error variance estimation updating.

`update` If not NULL, should be a list of starting values obtained from other optimization methods.

`useHsolnp` Whether to use the hessian matrix derived from the solnp optimization algorithm.

`...` Further arguments that will be passed to `modFit`.

Details

Instead of implicitly assuming equal error variances or giving arbitrary weights decided by the researcher as in the NLS algorithm, an iteratively reweighted least squares (IRLS) algorithm was implemented to obtain the maximum likelihood estimates of the kinetic model parameters.

Value

A list with "kingui", "mkinfit" and "modFit" in the class attribute. A summary can be obtained by `summary.kingui`.

Author(s)

Zhenglei Gao

Examples

```
complex <- mkinmod.full(
 parent = list(type = "SFO", to = c("A1", "B1", "C1"), sink = FALSE),
 A1 = list(type = "SFO", to = "A2"),
 B1 = list(type = "SFO"),
 C1 = list(type = "SFO"),
 A2 = list(type = "SFO"),
 inpartri='default',
 outpartri='default',
 data=schaefer07_complex_case,
 weight=NULL)
Fit    <- IRLSkinfit.full(
           complex,
              plot      = TRUE,
              quiet     = TRUE,
              ctr       = kingui.control
                           (method = 'solnp',
                           submethod = 'Port',
                             maxIter = 100,
                           tolerance = 1E-06,
                           odesolver = 'lsoda'),
           irls.control = list(
                             maxIter = 10,
                           tolerance = 0.001))
```

IRLSkinfit.gui *Fit a kinetic model using the IRLS algorithm.*

Description

This function does kinetic evaluations using the IRLS algorithm. This is deprecated! Use `IRLSkinfit.full` instead.

Usage

```
IRLSkinfit.gui(mkinmodini, eigen = FALSE, plot = FALSE,
  plottitle = "", quiet = FALSE, err = NULL,
  weight = "none", scaleVar = FALSE,
  ctr = kingui.control(), irls.control = list(),
  update = NULL, useHsolnp = FALSE, ...)
```

Arguments

`mkinmodini` A list of class `mkinmod.gui`, containing the kinetic model to be fitted to the data, and the initial parameter values, the observed data.

`eigen` If TRUE, the solution of the system of differential equations should be based on the spectral decomposition of the coefficient matrix in cases that this is possible.

`plot` If TRUE,the observed values and the numerical solutions should be plotted at each stage of the optimisation.

`plottitle` The title of the plot for visualizing the optimization process.

`quiet` If TRUE, suppress printing out the current model cost after each(>1) improvement.

`err` See argumetns of `mkinfit.gui`

`weight` See argumetns of `mkinfit.gui`

`scaleVar` See argumetns of `mkinfit.gui`

`ctr` a list of control values for the estimation algorithm to replace the default values including maximum iterations and absolute error tolerance. Defaults to the output of `kingui.control`.

`irls.control` A list of control values for the estimation algorithm to replace the default values including the maximum number of iterations for the outer iteration and the error tolerance level for the error variance estimation updating.

`update` NULL, should be a list of starting values obtained from other optimization methods.

`...` Further arguments that will be passed to `modFit1`.

Value

A list with "kingui", "mkinfit" and "modFit" in the class attribute. A summary can be obtained by `summary.kingui`.

Note

Deprecated!

Author(s)

Zhenglei Gao

See Also

`IRLSkinfit`, `mkinfit.gui`

Examples

```
SFO_SFO_gui <- mkinmod.gui(Parent = list(type = "SFO", to = "Metab", sink = TRUE,
                          k = list(ini = 0.1,
                     fixed = 0,
                     lower = 0,
                     upper = Inf),
              M0 = list(ini = 195,
                     fixed = 0,
                     lower = 0,
                     upper = Inf),
                          FF = list(ini = c(.1),
                     fixed = c(0),
                     lower = c(0),
                     upper = c(1)),
                     time=c(0.0,2.8,   6.2,  12.0,  29.2,  66.8,  99.8,
127.5, 154.4, 229.9, 272.3, 288.1, 322.9),
                    residue = c( 157.3, 206.3, 181.4, 223.0, 163.2,
144.7,  85.0,  76.5,  76.4,  51.5,  45.5,  47.3, 42.7)),
                          Metab = list(type = "SFO",
                          k = list(ini = 0.1   ,
                     fixed = 0,
                     lower = 0,
                     upper = Inf),
              M0 = list(ini = 0,
                     fixed = 1,
                     lower = 0,
                     upper = Inf),
                    residue =c( 0.0,  0.0,  0.0,  1.6,  4.0, 12.3, 13.5,
12.7, 11.4, 11.6, 10.9,  9.5,  7.6))                            )
fit <- IRLSkinfit.gui(SFO_SFO_gui,plot=T,ctr=kingui.control(maxIter=100,
tolerance=1e-6,odesolver='lsoda'))
```

---

kingraph *Function to generate graph data for the GUI to make fit plot.*

Description

A data table is generated and stored in file=filename for later usage.

Usage

```
kingraph(fit, filename = "graphdata.txt", xlab = "Time",
  ylab = "Observed",
  xlim = c(1, 1.05) * range(fit$data$time),
  ylim = c(1, 1.05) * range((summary(fit))$data$observed, na.rm = TRUE),
  legend = TRUE, length.out = 100, ...)
```

Arguments

fit	An object of class 'kingui'.
filename	The file in which the graph data will be stored.
xlim	The plot range for x axis.
ylim	The plot range for y axis.

Value

NULL

Note

This function only writes the data table for making the graph.

Author(s)

Zhenglei Gao

See Also

kinplot

---

kingui.control *Choose the algorithm to use and related control parameters in kinetic evaluations using NLS and IRLS.*

Description

Choose the algorithm to use and related control parameters in kinetic evaluations using NLS and IRLS.

Usage

```
kingui.control(method = "solnp", maxIter = 100,
  tolerance = 1e-08, odesolver = "lsoda", atol = 1e-09,
  rtol = 1e-10, rhobeg = 0.05, iprint = 1, trace = 0,
  goMarq = 0, delta = 1e-06, rho = 1, submethod = "Port",
  Hmethod1 = "L-BFGS-B", Hmethod2 = "Marq",
  quiet.tol = 1, ...)
```

Arguments

`method` The method to be used, one of "Rsolnp", "minqa","spg", "Marq", "Port", "Newton", "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Pseudo" - see details.

`maxIter` Numerber of maximum iterations

`tolerance` A positive numeric value specifying the tolerance level for the relative offset convergence criterion.

`odesolver` The integration routines used. see ode

`atol` absolute error tolerance, either a scalar or an array as long as 'y'. See ode

`rtol` Relative error tolerance, either a scalar or an array as long as 'y'.See ode.

`rhobeg` Initial trust region radius for method 'bobyqa'. See details of see bobyqa.

`iprint` Control the amount of printing by setting iprint to 0, 1 2, or 3. See details of see bobyqa.

`trace` A logical variable (TRUE/FALSE). If 'TRUE', information on the progress of optimization is printed. See details of see spg.

`goMarq` If TRUE, using "Marq" for the iterations after the first step in IRLS.

`delta` Control parameters for 'solnp'. See details of see solnp.

`rho` Control parameters for 'solnp'. See details of see solnp.

`submethod` If the method chosen failed to produce results, run the optimization using a substitute method.

`Hmethod1` The method used to calculate Hessian matrix

`Hmethod2`	The substitute method if `Hmethod1` fails.
`quiet.tol`	The improvement level at which the objective function(cost value) and the plot should be updated and reported.
`...`	Other characteristics of different optimizer for the users to play with.

Value

A list of control parameters for the ODE solver and the optimization routines.

Note

Should have a control function for `mcmckinfit.full`

Author(s)

Zhenglei Gao

Examples

```
kingui.control()
```

`KinGUI2-package` *Kinetic Evaluations in R*

Description

In-line documentation for R.

Details

Roxygen is a Doxygen-like documentation system for R; allowing in-source specification of Rd files, collation and namespace directives.

If you have existing Rd files, check out the `Rd2roxygen` package for a convenient way of converting Rd files to roxygen comments.

Package:	KinGUI2
Type:	Package
Version:	0.1-1
Date:	2012-03-15
License:	GPL (>= 2)
LazyLoad:	yes

Note

This is based on the package `mkin` by Johannes Reinken.

You are welcome to report other serious issues and bugs via `http://dropbox.com`.

Author(s)

Zhenglei Gao <`zhenglei.gao@bayer.com`>

Maintainer: Zhenglei Gao <zhenglei.gao@bayer.com>

Examples

```
## see the package vignette: vignette('KinGUI2')
```

---

kinplot *Plot the fit results*

Description

Function to plot the fit results for the whole model or selected compartments.

Usage

```
kinplot(fit, name = NULL, xlab = "Time",
  ylab = "Observed",
  xlim = c(1, 1.1) * range(fit$data$time),
  ylim = c(1, 1.1) * range(fit$data$observed, na.rm = TRUE),
  legend = TRUE, pdf = FALSE, ...)
```

Arguments

fit	An object of class 'kingui'.
name	Names of the compartmens that should be plotted.
xlab	xlab name.
ylab	ylab name.
xlim	The plot range for x axis.
ylim	The plot range for y axis.
legend	whether to include legends.
pdf	whether to put the plots into one pdf file.
...	optional arguments. Not used for now.

Value

NULL

Note

This function makes the plots.

Author(s)

Zhenglei Gao

See Also

kingraph

mcmckinfit.full *Fit a kinetic model using MCMC methods.*

Description

This function does kinetic evaluations using MCMC algorithm with function modMCMC in the FME package.

Usage

```
mcmckinfit.full(mkinmodini, eigen = FALSE,
  ctr = kingui.control(quiet.tol = 1e-05),
  plottitle = "", plot = FALSE, quiet = FALSE,
  err = NULL, weight = "none", scaleVar = FALSE,
  commonsigma = FALSE, jump = NULL, prior = NULL,
  wvar0 = 0.1, niter = 1000, outputlength = niter,
  burninlength = 0, updatecov = niter, ntrydr = 1,
  drscale = NULL, verbose = TRUE, fitstart = TRUE,
  update = NULL, ...)
```

Arguments

`mkinmodini` A list of class `mkinmod.full`, containing the kinetic model to be fitted to the data, and the initial parameter values, the observed data.

`eigen` If TRUE, the solution of the system of differential equations should be based on the spectral decomposition of the coefficient matrix in cases that this is possible.

`ctr` Used when fitstart is TRUE. A list of control values for the estimation algorithm to replac the default values including maximum iterations and absolute error tolerance. Defaults to the output of `kingui.control`.

`plottitle` The title of the plot for visualizing the optimization process.

`plot` When fitstart==TRUE, if TRUE,the observed values and the numerical solutions should be plotted at each stage of the first optimization step.

`quiet` If TRUE, suppress printing out the current model cost after each(>1) improvement.

`commonsigma` If TRUE,the error model has constant error variance and the NLS algorithm will be used for the first optimizatio step.

`jump` Jump length, either a number, a vector with length equal to the total number of parameters, a covariance matrix, or a function that takes as input the current values of the parameters and produces as output the perturbed parameters. See details of `modMCMC`.

`prior` Prior probability of the parameters, either a function of the parameters or 'NULL'; in the latter case a flat prior is used (i.e. all parameters are equally likely).

`wvar0` "weight" for the initial model variance. See details of `modMCMC`.

`niter` number of iterations for the MCMC.

`outputlength` number of iterations kept in the output.

`burninlength` number of discarded initial iterations.

`updatecov` number of iterations after which the parameter covariance matrix is (re)evaluated based on the parameters kept thus far, and used to update the MCMC jumps.

`ntrydr` maximal number of tries for the delayed rejection procedure.

`drscale` for each try during delayed rejection, the cholesky decomposition of the proposal matrix is scaled with this amount; if 'NULL', it is assumed to be 'c(0.2,0.25, 0.333, 0.333, ...)'

`verbose` if 'TRUE': prints extra output.

`fitstart` if 'TRUE': first perform an optimization step and using the fitted parameters as the starting values for MCMC.

`update` if not NULL, using the values in the update as the starting values for MCMC.

`...` Further arguments that will be passed to `modFit`.

Value

A list with "mcmckingui" and "modMCMC" in the class attribute. A summary can be obtained by `summary.mcmckingui`.

Note

`mcmckinfit.gui` is the deprecated version.

Author(s)

Zhenglei Gao

See Also

`modMCMC`

Examples

```
## Not run:
SFO_SFO_gui <- mkinmod.gui(Parent = list(type = "SFO", to = "Metab", sink = TRUE,
                           k = list(ini = 0.1,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
              M0 = list(ini = 195,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
                           FF = list(ini = c(.1),
                      fixed = c(0),
                      lower = c(0),
                      upper = c(1)),
                      time=c(0.0,2.8,   6.2,  12.0,  29.2,  66.8,  99.8,
127.5, 154.4, 229.9, 272.3, 288.1, 322.9),
                    residue = c( 157.3, 206.3, 181.4, 223.0, 163.2,
144.7,  85.0,  76.5,  76.4,  51.5,  45.5,  47.3, 42.7)),
                           Metab = list(type = "SFO",
                           k = list(ini = 0.1    ,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
              M0 = list(ini = 0,
                      fixed = 1,
                      lower = 0,
                      upper = Inf),
                    residue =c( 0.0,  0.0,  0.0,  1.6,  4.0, 12.3, 13.5,
12.7, 11.4, 11.6, 10.9,  9.5,  7.6))                           )

fit <- mcmckinfit.gui(SFO_SFO_gui)

summary(fit)
plot(fit)

## End(Not run)
```

mkinfit.full *Fit a kinetic model using the NLS or WNLS algorithm.*

Description

KinGUI2 version of mkinfit. This function uses the Flexible Modelling Environment package FME to create a function calculating the model cost with weigths, which is then minimised , using the specified initial or fixed parameters and starting values. This is deprecated! Use mkinfit.full instead.

Usage

```
mkinfit.full(mkinmodini, eigen = FALSE, plot = FALSE,
  plottitle = "", quiet = FALSE, err = NULL,
  weight = "none", scaleVar = FALSE,
  ctr = kingui.control(), update = NULL, ...)
```

Arguments

mkinmodini	A list of class mkinmod.gui, containing the kinetic model to be fitted to the data, and the initial parameter values, the observed data.
eigen	If TRUE, the solution of the system of differential equations should be based on the spectral decomposition of the coefficient matrix in cases that this is possible.
plot	If TRUE,the observed values and the numerical solutions should be plotted at each stage of the optimization.
plottitle	The title of the plot for visualizing the optimization process.
quiet	If TRUE, suppress printing out the current model cost after each(>1) improvement.
err	Either NULL, or the name of the column with the *error* estimates, used to weigh the residuals (see details of modCost); if NULL, then the residuals are not weighed. In the GUI version, there is no need to consider this argument since a default weight one matrix is setup in mkinmod.gui. The err argument turned into 'err' automatically in the codes.
weight	only if err=NULL: how to weigh the residuals, one of "none", "std", "mean", see details of modCost.

`scaleVar` Will be passed to `modCost`. Default is not to scale Variables according to the number of observations.

`ctr` A list of control values for the estimation algorithm to replace the default values including maximum iterations and absolute error tolerance. Defaults to the output of `kingui.control`.

`...` Further arguments that will be passed to `modFit`.

Value

A list with "kingui", "mkinfit" and "modFit" in the class attribute. A summary can be obtained by `summary.kingui`.

Author(s)

Zhenglei Gao

See Also

`mkinfit`

Examples

```
## Not run:
guitest <- mkinmod.full(
    parent = list(
      time = c(     0,      3,      7,     14,     30,     62,     90,    118),
   residue = c(101.24,  99.27,  90.11,  72.19,  29.71,   5.98,   1.54,  NA),
    weight = c(     1,      1,      1,      1,      1,      1,      1,      1),
                  sink  = TRUE,
      type = "SFO",
         k = list(ini   = 0.040,
                  fixed = 0,
                  lower = 0.0,
                  upper = Inf),
        M0 = list(ini   = 100.15,
                  fixed = 0,
                  lower = 0.0,
                  upper = Inf)),
         inpartri=’default’,outpartri=’default’ )
 Fit    <- IRLSkinfit.full(
           guitest,
              plot      = TRUE,
              quiet     = TRUE,
             ctr       = kingui.control(
                              method = ’solnp’,
                           submethod = ’Port’,
                             maxIter = 100,
                           tolerance = 1E-06,
                           odesolver = ’lsoda’),
           irls.control = list(
                             maxIter = 10,
                           tolerance = 0.001))

## End(Not run)
```

mkinfit.gui *Fit a kinetic model using the NLS or WNLS algorithm.*

Description

GUI version of mkinfit. This function uses the Flexible Modelling Environment package FME to create a function calculating the model cost with weigths, which is then minimised , using the specified initial or fixed parameters and starting values. This is deprecated! Use mkinfit.full instead.

Usage

```
mkinfit.gui(mkinmodini, eigen = FALSE, plot = FALSE,
  plottitle = "", quiet = FALSE, err = NULL,
  weight = "none", scaleVar = FALSE,
  ctr = kingui.control(), ...)
```

Arguments

`mkinmodini`	A list of class `mkinmod.gui`, containing the kinetic model to be fitted to the data, and the initial parameter values, the observed data.
`eigen`	If TRUE, the solution of the system of differential equations should be based on the spectral decomposition of the coefficient matrix in cases that this is possible.
`plot`	If TRUE,the observed values and the numerical solutions should be plotted at each stage of the optimization.
`plottitle`	The title of the plot for visualizing the optimization process.
`quiet`	If TRUE, suppress printing out the current model cost after each(>1) improvement.
`err`	Either `NULL`, or the name of the column with the *error* estimates, used to weigh the residuals (see details of `modCost`); if `NULL`, then the residuals are not weighed. In the GUI version, there is no need to consider this argument since a default weight one matrix is setup in `mkinmod.gui`. The err argument turned into 'err' automatically in the codes.
`weight`	only if `err=NULL`: how to weigh the residuals, one of "none", "std", "mean", see details of `modCost`.
`scaleVar`	Will be passed to `modCost`. Default is not to scale Variables according to the number of observations.
`ctr`	A list of control values for the estimation algorithm to replace the default values including maximum iterations and absolute error tolerance. Defaults to the output of `kingui.control`.
`...`	Further arguments that will be passed to `modFit`.

Value

A list with "kingui", "mkinfit" and "modFit" in the class attribute. A summary can be obtained by `summary.kingui`.

Note

Deprecated!

Author(s)

Zhenglei Gao

See Also

mkinfit

Examples

```
## Not run:
SFO_SFO_gui <- mkinmod.gui(Parent = list(type = "SFO", to = "Metab", sink = TRUE,
                           k = list(ini = 0.1,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
              M0 = list(ini = 195,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
                           FF = list(ini = c(.1),
                      fixed = c(0),
                      lower = c(0),
                      upper = c(1)),
                      time=c(0.0,2.8,   6.2,  12.0,  29.2,  66.8,  99.8,
127.5, 154.4, 229.9, 272.3, 288.1, 322.9),
                    residue = c( 157.3, 206.3, 181.4, 223.0, 163.2,
144.7,  85.0,  76.5,  76.4,  51.5,  45.5,  47.3, 42.7)),
                           Metab = list(type = "SFO",
                           k = list(ini = 0.1   ,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
              M0 = list(ini = 0,
                      fixed = 1,
                      lower = 0,
                      upper = Inf),
                    residue =c( 0.0,  0.0,  0.0,  1.6,  4.0, 12.3, 13.5,
12.7, 11.4, 11.6, 10.9,  9.5,  7.6))                           )
fit <- mkinfit.gui(SFO_SFO_gui,plot=T,ctr=kingui.control(maxIter=100,
tolerance=1e-6,odesolver='lsoda'))

## End(Not run)
```

---

mkinmod.full *Full function to set up a kinetic model with one or more compartments and the data.*

Description

GUI version of mkinmod. The function takes a specification, consisting of a list of the compartments in the data. Each compartment is again represented by a list, specifying the kinetic model type, reaction or transfer to other observed compartments, the initial parameter values, lower and upper bounds, fixed or not, and observed data.

Usage

```
mkinmod.full(...,
  inpartri = c("default", "water-sediment", "advanced"),
  outpartri = c("default", "water-sediment", "advanced"),
  data = NULL, weight = NULL)
```

Arguments

`...` Each list cell represents a comparment which contains a list of comonents including 'type'(kinetic reaction type, single first order kinetics "SFO" are implemented for all compartments, while "FOMC", "DFOP" and "HS" can additionally be chosen for the first variable which is assumed to be the source compartment), each parameter name(a list of 'ini','fixed','lower','upper'),'residue'(measured concentrations),'time'(sampling time),'weight'(weights to be used, defaul 1),'sink'(Default TRUE, tranformation to unspecified compartments.),'to'(a vector of compartment names that the source compartment will be transferred to).

`inpartri` Input parameterization.

`outpartri` Output parameterization.

`data` Optional. Data can be read from a data file.

`weight` Optional. General weighting schemes using a weight matrix.

Value

A list of class 'mkinmod.full' for use with mkinfit.full,IRLSkinfit.full and mcmckinfit.full containing:

`diffs` A vector of string representations of differential equations, one for each modelling compartment.

`parms` A vector of parameter names occurring in the differential equations.

`map` A list containing named character vectors for each compartments in the model.

`parms.ini` Initial values for all kinetic parameters in the model.

`state.ini` Initial state values for all compartments in the model.

`lower` Lower bounds for the parameters(including state variables) to be optimized.

`upper` upper bounds for the parameters(including state variables) to be optimized.

`fixed_parms` The names of the kinetic parameters that are fixed during optimization.

`fixed_initials` The names of the initial states that are fixed during optimization.

`residue` The observed data matrix with a time column.

`weightmat` The weights matrix.

`ff` A vector of string representations of the transformation between the formation fractions in the model and the transfomed formation fractions in the optimization process.

Note

mkinmod is a deprecated version doing the same for the use with mkinfit.gui,IRLSkinfit.gui and mcmckinfit.gui.

Author(s)

Zhenglei Gao

See Also

mkinmod, CompleteCompound

Examples

```
SFO_SFO_full <- mkinmod.full(Parent = list(type = "SFO", to = "Metab", sink = TRUE,
                           k = list(ini = 0.1,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
              M0 = list(ini = 195,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
                           FF = list(ini = c(.1),
                      fixed = c(0),
                      lower = c(0),
                      upper = c(1)),
                      time=c(0.0,2.8,   6.2,  12.0,  29.2,  66.8,  99.8,
127.5, 154.4, 229.9, 272.3, 288.1, 322.9),
                    residue = c( 157.3, 206.3, 181.4, 223.0, 163.2,
144.7,  85.0,  76.5,  76.4,  51.5,  45.5,  47.3, 42.7)),
                           Metab = list(type = "SFO",
                           k = list(ini = 0.1    ,
                      fixed = 0,
                      lower = 0,
                      upper = Inf),
              M0 = list(ini = 0,
                      fixed = 1,
                      lower = 0,
                      upper = Inf),
                    residue =c( 0.0,  0.0,  0.0,  1.6,  4.0, 12.3, 13.5,
12.7, 11.4, 11.6, 10.9,  9.5,  7.6))
```

---

plot.mcmckingui *S3 plot method to plot for calss 'mcmckingui'*

Description

Make the density correlation and trace plots of the generated Markov Chains.

Usage

```
plot.mcmckingui(object, fname1 = "density",
  fname2 = "correlation", fname3 = "trace", pch = 1,
  device = "wmf", ...)
```

Arguments

`object`	An object of class 'mcmckingui'
`fname1`	The file name of the density plot.
`fname2`	The file name of the correlation plot.
`pch`	What kind of points to use in the plots.
`device`	The plot device to be used.
`...`	Other arguments to be passed to 'plot'.

Value

Density and Correlation plots of the sampled parameters in 'wmf' or other format.

Note

`plot.mcmckingui0` is a deprecated version to be used with the object returned by `mcmckinfit.gui`

Author(s)

Zhenglei Gao

`print.summary.kingui` *Print method for* `summary.kingui`

Description

Print method for `summary.kingui`

Usage

```
summary(Fit))
```

Arguments

`x`	An object of class `summary.kingui`
`digits`	How many digits should be printed after the decimal point.
`detailed`	Ignore for now.
`...`	Other parameters to be passed into `summary.kingui`

Details

Expanded from `print.summary.modFit` and `print.summary.mkinfit`

Value

`NULL`

Author(s)

Zhenglei Gao

print.summary.mcmckingui *S3 print method for class* `summary.mcmckingui`

Description

S3 print method for class `summary.mcmckingui`

Usage

```
print.summary.mcmckingui(x,
  digits = max(3, getOption("digits") - 3),
  detailed = FALSE, ...)
```

Arguments

`x`	An object of class `summary.mcmckingui`.
`digits`	Number of digits to be printed after the decimal point.
`detailed`	Not used.
`...`	Optional arguments.

Value

`NULL`

Author(s)

Zhenglei Gao

rmse *Calculating the root mean squared error.*

Description

Calculating the scaled root mean squared error according to Focus Kinetics Guidelines.

Usage

```
rmse(obs, pred)
```

Arguments

`obs`	Observed values.
`pred`	Predicted values using model.

Value

The root mean squared error.

Author(s)

Zhenglei Gao

`summary.kingui` *S3 summary method for class 'kingui'*

Description

Lists model equations, the chi2 error levels calculated according to FOCUS guidance (2006) and optionally the data, consisting of observed, predicted and residual values, the correlation matrix and so on.

Usage

```
summary(object, data = TRUE, distimes = TRUE, ff = TRUE,
  cov = FALSE,version="1.2011 .922.1530",...)
```

Arguments

`object`	A fitted object of class 'kingui' from the result of `mkinfit.full` or `IRLSkinfit.full`
`data`	If TRUE, include in the returned values a data frame containing the observed and predicted values with residuals and estimated standard deviations or weights.
`distimes`	If TRUE, DT50 and DT90 values should be included.
`ff`	If TRUE, the formation fraction should be calculated from the estimated transformed parameters.
`cov`	If TRUE, parameter covariances should be calculated.
`version`	A version number indicating which version of the fit function has been used.
`...`	Optional arguments passed to methods like 'print'

Details

Expanded from `summary.modFit` and `summary.mkinfit`

Value

The summary function returns a list of components from the results of the optimizat ion routine used.

`par`	The fitted parameter values with corresponding standard error and t-test p values.
`version`	The version of function used.
`pnames`	Parameter names.
`diffs`	The differential equations used in the model.
`data`	A data frame that will be printed.
`start`	The starting values and bounds for the parameters.
`fixed`	The fixed values for model parameters.
`start0`	The starting values for the transformed parameters.
`fixed0`	The fixed values for the transformed parameters.

`errmin` Chi2 error level and other related statistics for goodness-of-fit assessment.

`distimes` The DT50 and DT90 values.

`ff` The formation fractions calculated from the fitted parameters.

`outpartri` Output parameterization.

`optimmethod` Optimization method used in the final fitting step and in calculating Hessian.

`residuals` The residuals.

`residualVariance`
The variance of the residuals.

`sigma` The standard deviation of the residuals.

`df` Degree of freedom for the residuals

`cov.unscaled` Unscaled Covariance

`cov.scaled` Scaled Covariance.

`info` Information inherated from the fitting procedure

`niter` Number of iterations.

`stopmess` Warning message.

Author(s)

Zhenglei Gao

`summary.mcmckingui` *S3 summary method for class* `mcmckingui`

Description

Lists model equations, the chi2 error levels calculated according to FOCUS guidance (2006) and optionally the data, consisting of observed, predicted and residual values, the correlation matrix and also the same summary statistics for class `modMCMC`

Usage

```
summary(object, remove = NULL,
  data=TRUE,distimes=TRUE,ff=TRUE,
  version=”1.2011.922.1530”,...)
```

Arguments

`object` A fitted object of class `mcmckingui` from the result of `mcmckinfit.full`.

`remove` The iterations should be removed from summary statistics calculations.

`data` If TRUE, include in the returned values a data frame containing the observed and predicted values with residuals and estimated standard deviations or weights.

`distimes` If TRUE, DT50 and DT90 values should be included.

`ff` If TRUE, the formation fraction should be calculated from the estimated transformed parameters.

`version` A version number indicating which version of the fit function has been used.

`...` Optional arguments passed to methods like `print`.

Value

The summary function returns a list of components from the results of the optimizat ion routine used.

Author(s)

Zhenglei Gao

See Also

`summary.kingui`.

Index