Package 'mkin'

November 16, 2017

Type Package

Title Kinetic Evaluation of Chemical Degradation Data

Version 0.9.46.3

Date 2017-11-16

Description Calculation routines based on the FOCUS Kinetics Report (2006, 2014). Includes a function for conveniently defining differential equation models, model solution based on eigenvalues if possible or using numerical solvers and a choice of the optimisation methods made available by the 'FME' package. If a C compiler (on windows: 'Rtools') is installed, differential equation models are solved using compiled C functions. Please note that no warranty is implied for correctness of results or fitness for a particular purpose.

Imports stats, graphics, methods, FME, deSolve, R6, minpack.lm, rootSolve, inline, parallel

Suggests knitr, rbenchmark, tikzDevice, testthat

License GPL

LazyLoad yes

LazyData yes

Encoding UTF-8

VignetteBuilder knitr

BugReports http://github.com/jranke/mkin/issues

URL https://cgit.jrwb.de/mkin/about

NeedsCompilation no

Author Johannes Ranke [aut, cre, cph] (0000-0003-4371-6538),
Katrin Lindenberger [ctb],
René Lehmann [ctb],
Eurofins Regulatory AG [cph]

Maintainer Johannes Ranke <jranke@uni-bremen.de>

Repository CRAN

Date/Publication 2017-11-16 19:33:51 UTC

R topics documented:

add_err *Add normally distributed errors to simulated kinetic degradation data*

Description

Normally distributed errors are added to data predicted for a specific degradation model using mkinpredict. The variance of the error may depend on the predicted value and is specified as a standard deviation.

Usage

```
add_err(prediction, sdfunc, secondary = c("M1", "M2"),
        n = 1000, LOD = 0.1, reps = 2,
        digits = 1, seed = NA)
```

Arguments

prediction	A prediction from a kinetic model as produced by mkinpredict.
sdfunc	A function taking the predicted value as its only argument and returning a standard deviation that should be used for generating the random error terms for this value.
secondary	The names of state variables that should have an initial value of zero
n	The number of datasets to be generated.
LOD	The limit of detection (LOD). Values that are below the LOD after adding the random error will be set to NA.
reps	The number of replicates to be generated within the datasets.
digits	The number of digits to which the values will be rounded.
seed	The seed used for the generation of random numbers. If NA, the seed is not set.

Value

A list of datasets compatible with mmkin, i.e. the components of the list are datasets compatible with mkinfit.

Author(s)

Johannes Ranke

References

Ranke J and Lehmann R (2015) To t-test or not to t-test, that is the question. XV Symposium on Pesticide Chemistry 2-4 September 2015, Piacenza, Italy http://chem.uft.uni-bremen.de/ranke/posters/piacenza_2015.pdf

Examples

```
# The kinetic model
m_SFO_SFO <- mkinmod(parent = mkinsub("SFO", "M1"),
                     M1 = mkinsub("SFO"), use_of_ff = "max")

# Generate a prediction for a specific set of parameters
sampling_times = c(0, 1, 3, 7, 14, 28, 60, 90, 120)

# This is the prediction used for the "Type 2 datasets" on the Piacenza poster
# from 2015
d_SFO_SFO <- mkinpredict(m_SFO_SFO,
                         c(k_parent = 0.1, f_parent_to_M1 = 0.5,
                           k_M1 = log(2)/1000),
                         c(parent = 100, M1 = 0),
                         sampling_times)

# Add an error term with a constant (independent of the value) standard deviation
# of 10, and generate three datasets
d_SFO_SFO_err <- add_err(d_SFO_SFO, function(x) 10, n = 3, seed = 123456789 )

# Name the datasets for nicer plotting
names(d_SFO_SFO_err) <- paste("Dataset", 1:3)

# Name the model in the list of models (with only one member in this case)
# for nicer plotting later on.
# Be quiet and use the faster Levenberg-Marquardt algorithm, as the datasets
# are easy and examples are run often. Use only one core not to offend CRAN
# checks
f_SFO_SFO <- mmkin(list("SFO-SFO" = m_SFO_SFO),
                   d_SFO_SFO_err, cores = 1,
                   quiet = TRUE, method.modFit = "Marq")

plot(f_SFO_SFO)

# We would like to inspect the fit for dataset 3 more closely
# Using double brackets makes the returned object an mkinfit object
# instead of a list of mkinfit objects, so plot.mkinfit is used
plot(f_SFO_SFO[[3]], show_residuals = TRUE)

# If we use single brackets, we should give two indices (model and dataset),
# and plot.mmkin is used
plot(f_SFO_SFO[1, 3])
```

`DFOP.solution` *Double First-Order in Parallel kinetics*

Description

Function describing decline from a defined starting value using the sum of two exponential decline functions.

Usage

```
DFOP.solution(t, parent.0, k1, k2, g)
```

Arguments

t	Time.
parent.0	Starting value for the response variable at time zero.
k1	First kinetic constant.
k2	Second kinetic constant.
g	Fraction of the starting value declining according to the first kinetic constant.

Value

The value of the response variable at time t.

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
plot(function(x) DFOP.solution(x, 100, 5, 0.5, 0.3), 0, 4, ylim=c(0,100))
```

---

endpoints — *Function to calculate endpoints for further use from kinetic models fitted with mkinfit*

Description

This function calculates DT50 and DT90 values as well as formation fractions from kinetic models fitted with mkinfit. If the SFORB model was specified for one of the parents or metabolites, the Eigenvalues are returned. These are equivalent to the rate constantes of the DFOP model, but with the advantage that the SFORB model can also be used for metabolites.

Usage

```
endpoints(fit)
```

Arguments

fit	An object of class mkinfit.

Value

A list with the components mentioned above.

Note

The function is used internally by `summary.mkinfit`.

Author(s)

Johannes Ranke

Examples

```
fit <- mkinfit("FOMC", FOCUS_2006_C, quiet = TRUE)
endpoints(fit)
```

`FOCUS_2006_datasets` *Datasets A to F from the FOCUS Kinetics report from 2006*

Description

Data taken from FOCUS (2006), p. 258.

Usage

```
FOCUS_2006_datasets
```

Format

6 datasets with observations on the following variables.

`name` a factor containing the name of the observed variable

`time` a numeric vector containing time points

`value` a numeric vector containing concentrations in percent of applied radioactivity

Source

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
FOCUS_2006_C
```

FOCUS_2006_DFOP_ref_A_to_B

Results of fitting the DFOP model to Datasets A to B of FOCUS (2006)

Description

A table with the fitted parameters and the resulting DT50 and DT90 values generated with different software packages. Taken directly from FOCUS (2006). The results from fitting the data with the Topfit software was removed, as the initial concentration of the parent compound was fixed to a value of 100 in this fit.

Usage

```
data(FOCUS_2006_DFOP_ref_A_to_B)
```

Format

A data frame containing the following variables.

`package` a factor giving the name of the software package

`M0` The fitted initial concentration of the parent compound

`f` The fitted f parameter

`k1` The fitted k1 parameter

`k2` The fitted k2 parameter

`DT50` The resulting half-life of the parent compound

`DT90` The resulting DT90 of the parent compound

`dataset` The FOCUS dataset that was used

Source

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
data(FOCUS_2006_DFOP_ref_A_to_B)
```

FOCUS_2006_FOMC_ref_A_to_F

Results of fitting the FOMC model to Datasets A to F of FOCUS (2006)

Description

A table with the fitted parameters and the resulting DT50 and DT90 values generated with different software packages. Taken directly from FOCUS (2006). The results from fitting the data with the Topfit software was removed, as the initial concentration of the parent compound was fixed to a value of 100 in this fit.

Usage

```
data(FOCUS_2006_FOMC_ref_A_to_F)
```

Format

A data frame containing the following variables.

`package` a factor giving the name of the software package

`M0` The fitted initial concentration of the parent compound

`alpha` The fitted alpha parameter

`beta` The fitted beta parameter

`DT50` The resulting half-life of the parent compound

`DT90` The resulting DT90 of the parent compound

`dataset` The FOCUS dataset that was used

Source

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
data(FOCUS_2006_FOMC_ref_A_to_F)
```

FOCUS_2006_HS_ref_A_to_F *Results of fitting the HS model to Datasets A to F of FOCUS (2006)*

Description

A table with the fitted parameters and the resulting DT50 and DT90 values generated with different software packages. Taken directly from FOCUS (2006). The results from fitting the data with the Topfit software was removed, as the initial concentration of the parent compound was fixed to a value of 100 in this fit.

Usage

```
data(FOCUS_2006_HS_ref_A_to_F)
```

Format

A data frame containing the following variables.

`package` a factor giving the name of the software package

`M0` The fitted initial concentration of the parent compound

`tb` The fitted tb parameter

`k1` The fitted k1 parameter

`k2` The fitted k2 parameter

`DT50` The resulting half-life of the parent compound

`DT90` The resulting DT90 of the parent compound

`dataset` The FOCUS dataset that was used

Source

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
data(FOCUS_2006_HS_ref_A_to_F)
```

FOCUS_2006_SFO_ref_A_to_F *Results of fitting the SFO model to Datasets A to F of FOCUS (2006)*

Description

A table with the fitted parameters and the resulting DT50 and DT90 values generated with different software packages. Taken directly from FOCUS (2006). The results from fitting the data with the Topfit software was removed, as the initial concentration of the parent compound was fixed to a value of 100 in this fit.

Usage

```
data(FOCUS_2006_SFO_ref_A_to_F)
```

Format

A data frame containing the following variables.

`package` a factor giving the name of the software package

`M0` The fitted initial concentration of the parent compound

`k` The fitted first-order degradation rate constant

`DT50` The resulting half-life of the parent compound

`DT90` The resulting DT90 of the parent compound

`dataset` The FOCUS dataset that was used

Source

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
data(FOCUS_2006_SFO_ref_A_to_F)
```

FOMC.solution *First-Order Multi-Compartment kinetics*

Description

Function describing exponential decline from a defined starting value, with a decreasing rate constant.

The form given here differs slightly from the original reference by Gustafson and Holden (1990). The parameter beta corresponds to 1/beta in the original equation.

Usage

```
FOMC.solution(t, parent.0, alpha, beta)
```

Arguments

t	Time.
parent.0	Starting value for the response variable at time zero.
alpha	Shape parameter determined by coefficient of variation of rate constant values.
beta	Location parameter.

Value

The value of the response variable at time t.

Note

The solution of the FOMC kinetic model reduces to the SFO.solution for large values of alpha and beta with $k = \frac{\beta}{\alpha}$.

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Gustafson DI and Holden LR (1990) Nonlinear pesticide dissipation in soil: A new model based on spatial variability. *Environmental Science and Technology* **24**, 1032-1038

Examples

```
plot(function(x) FOMC.solution(x, 100, 10, 2), 0, 2, ylim = c(0, 100))
```

geometric_mean *Calculate the geometric mean*

Description

Function calculating the geometric mean of numeric vectors

Usage

```
geometric_mean(x, na.rm = FALSE)
```

Arguments

x	A numeric vector
na.rm	Should NA values be ignored

Value

The geometric mean.

Examples

```
geometric_mean(c(1,3, 9))
geometric_mean(c(1,3, NA))
geometric_mean(c(1,3, NA), na.rm = TRUE)
```

HS.solution *Hockey-Stick kinetics*

Description

Function describing two exponential decline functions with a break point between them.

Usage

```
HS.solution(t, parent.0, k1, k2, tb)
```

Arguments

t	Time.
parent.0	Starting value for the response variable at time zero.
k1	First kinetic constant.
k2	Second kinetic constant.
tb	Break point. Before this time, exponential decline according to k1 is calculated, after this time, exponential decline proceeds according to k2.

Value

The value of the response variable at time t.

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
plot(function(x) HS.solution(x, 100, 2, 0.3, 0.5), 0, 2, ylim=c(0,100))
```

---

ilr *Function to perform isometric log-ratio transformation*

Description

This implementation is a special case of the class of isometric log-ratio transformations.

Usage

```
ilr(x)
invilr(x)
```

Arguments

x	A numeric vector. Naturally, the forward transformation is only sensible for vectors with all elements being greater than zero.

Value

The result of the forward or backward transformation. The returned components always sum to 1 for the case of the inverse log-ratio transformation.

Author(s)

René Lehmann and Johannes Ranke

References

Peter Filzmoser, Karel Hron (2008) Outlier Detection for Compositional Data Using Robust Methods. Math Geosci 40 233-248

See Also

Another implementation can be found in R package robCompositions.

Examples

```
# Order matters
ilr(c(0.1, 1, 10))
ilr(c(10, 1, 0.1))
# Equal entries give ilr transformations with zeros as elements
ilr(c(3, 3, 3))
# Almost equal entries give small numbers
ilr(c(0.3, 0.4, 0.3))
# Only the ratio between the numbers counts, not their sum
invilr(ilr(c(0.7, 0.29, 0.01)))
invilr(ilr(2.1 * c(0.7, 0.29, 0.01)))
# Inverse transformation of larger numbers gives unequal elements
invilr(-10)
invilr(c(-10, 0))
# The sum of the elements of the inverse ilr is 1
sum(invilr(c(-10, 0)))
# This is why we do not need all elements of the inverse transformation to go back:
a <- c(0.1, 0.3, 0.5)
b <- invilr(a)
length(b) # Four elements
ilr(c(b[1:3], 1 - sum(b[1:3]))) # Gives c(0.1, 0.3, 0.5)
```

IORE.solution *Indeterminate order rate equation kinetics*

Description

Function describing exponential decline from a defined starting value, with a concentration dependent rate constant.

Usage

```
IORE.solution(t, parent.0, k__iore, N)
```

Arguments

t	Time.
parent.0	Starting value for the response variable at time zero.
k__iore	Rate constant. Note that this depends on the concentration units used.
N	Exponent describing the nonlinearity of the rate equation

Value

The value of the response variable at time t.

Note

The solution of the IORE kinetic model reduces to the SFO.solution if N = 1. The parameters of the IORE model can be transformed to equivalent parameters of the FOMC mode - see the NAFTA guidance for details.

References

NAFTA Technical Working Group on Pesticides (not dated) Guidance for Evaluating and Calculating Degradation Kinetics in Environmental Media

Examples

```
plot(function(x) IORE.solution(x, 100, 0.2, 1.3), 0, 2,
                              ylim = c(0, 100))
fit.fomc <- mkinfit("FOMC", FOCUS_2006_C, quiet = TRUE)
fit.iore <- mkinfit("IORE", FOCUS_2006_C, quiet = TRUE)
fit.iore.deS <- mkinfit("IORE", FOCUS_2006_C, solution_type = "deSolve", quiet = TRUE)

print(data.frame(coef(fit.fomc), coef(fit.iore), coef(fit.iore.deS),
                 row.names = paste("model par", 1:3)))
print(rbind(fomc = endpoints(fit.fomc)$distimes, iore = endpoints(fit.iore)$distimes,
            iore.deS = endpoints(fit.iore)$distimes))
```

---

max_twa_parent *Function to calculate maximum time weighted average concentrations from kinetic models fitted with mkinfit*

Description

This function calculates maximum moving window time weighted average concentrations (TWAs) for kinetic models fitted with mkinfit. Currently, only calculations for the parent are implemented for the SFO, FOMC and DFOP models, using the analytical formulas given in the PEC soil section of the FOCUS guidance.

Usage

```
max_twa_parent(fit, windows)
```

Arguments

fit	An object of class mkinfit.
windows	The width of the time windows for which the TWAs should be calculated.

Value

A numeric vector, named using the windows argument.

Author(s)

Johannes Ranke

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
fit <- mkinfit("FOMC", FOCUS_2006_C, quiet = TRUE)
max_twa_parent(fit, c(7, 21))
```

---

mccall81_245T *Datasets on aerobic soil metabolism of 2,4,5-T in six soils*

Description

Time course of 2,4,5-trichlorophenoxyacetic acid, and the corresponding 2,4,5-trichlorophenol and 2,4,5-trichloroanisole as recovered in diethylether extracts.

Usage

```
mccall81_245T
```

Format

A dataframe containing the following variables.

`name` the name of the compound observed. Note that T245 is used as an acronym for 2,4,5-T. T245 is a legitimate object name in R, which is necessary for specifying models using `mkinmod`.

`time` a numeric vector containing sampling times in days after treatment

`value` a numeric vector containing concentrations in percent of applied radioactivity

`soil` a factor containing the name of the soil

Source

McCall P, Vrona SA, Kelley SS (1981) Fate of uniformly carbon-14 ring labeled 2,4,5-Trichlorophenoxyacetic acid and 2,4-dichlorophenoxyacetic acid. J Agric Chem 29, 100-107 http://dx.doi.org/10.1021/jf00103a026

Examples

```
SFO_SFO_SFO <- mkinmod(T245 = list(type = "SFO", to = "phenol"),
                       phenol = list(type = "SFO", to = "anisole"),
                       anisole = list(type = "SFO"))
## Not run:
 fit.1 <- mkinfit(SFO_SFO_SFO, subset(mccall81_245T, soil == "Commerce"), quiet = TRUE)
  summary(fit.1, data = FALSE)

## End(Not run)
  # No convergence, no covariance matrix ...
  # k_phenol_sink is really small, therefore fix it to zero
  fit.2 <- mkinfit(SFO_SFO_SFO, subset(mccall81_245T, soil == "Commerce"),
                parms.ini = c(k_phenol_sink = 0),
                fixed_parms = "k_phenol_sink", quiet = TRUE)
  summary(fit.2, data = FALSE)
```

---

mkinds *A dataset class for mkin*

Description

A dataset class for mkin

Usage

```
mkinds
```

Format

An R6Class generator object.

Fields

`title` A full title for the dataset

`sampling` times The sampling times

`time_unit` The time unit

`observed` Names of the observed compounds

`unit` The unit of the observations

`replicates` The number of replicates

`data` A dataframe with at least the columns name, time and value in order to be compatible with mkinfit

Examples

```
mds <- mkinds$new("FOCUS A", FOCUS_2006_A)
```

`mkinerrmin` *Calculate the minimum error to assume in order to pass the variance test*

Description

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

Usage

```
mkinerrmin(fit, alpha = 0.05)
```

Arguments

`fit`	an object of class `mkinfit`.
`alpha`	The confidence level chosen for the chi-squared test.

Details

This function is used internally by `summary.mkinfit`.

Value

A dataframe with the following components:

`err.min`	The relative error, expressed as a fraction.
`n.optim`	The number of optimised parameters attributed to the data series.
`df`	The number of remaining degrees of freedom for the chi2 error level calculations. Note that mean values are used for the chi2 statistic and therefore every time point with observed values in the series only counts one time.

The dataframe has one row for the total dataset and one further row for each observed state variable in the model.

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
SFO_SFO = mkinmod(parent = mkinsub("SFO", to = "m1"),
                  m1 = mkinsub("SFO"),
                  use_of_ff = "max")

fit_FOCUS_D = mkinfit(SFO_SFO, FOCUS_2006_D, quiet = TRUE)
round(mkinerrmin(fit_FOCUS_D), 4)
fit_FOCUS_E = mkinfit(SFO_SFO, FOCUS_2006_E, quiet = TRUE)
round(mkinerrmin(fit_FOCUS_E), 4)
```

---

mkinfit *Fit a kinetic model to data with one or more state variables*

Description

This function uses the Flexible Modelling Environment package FME to create a function calculating the model cost, i.e. the deviation between the kinetic model and the observed data. This model cost is then minimised using the Port algorithm nlminb, using the specified initial or fixed parameters and starting values. Per default, parameters in the kinetic models are internally transformed in order to better satisfy the assumption of a normal distribution of their estimators. In each step of the optimsation, the kinetic model is solved using the function mkinpredict. The variance of the residuals for each observed variable can optionally be iteratively reweighted until convergence using the argument `reweight.method = "obs"`.

Usage

```
mkinfit(mkinmod, observed,
  parms.ini = "auto",
  state.ini = "auto",
  fixed_parms = NULL, fixed_initials = names(mkinmod$diffs)[-1],
  from_max_mean = FALSE,
  solution_type = c("auto", "analytical", "eigen", "deSolve"),
  method.ode = "lsoda",
  use_compiled = "auto",
  method.modFit = c("Port", "Marq", "SANN", "Nelder-Mead", "BFGS", "CG", "L-BFGS-B"),
  maxit.modFit = "auto",
  control.modFit = list(),
  transform_rates = TRUE,
  transform_fractions = TRUE,
  plot = FALSE, quiet = FALSE, err = NULL, weight = "none",
  scaleVar = FALSE,
  atol = 1e-8, rtol = 1e-10, n.outtimes = 100,
  reweight.method = NULL,
  reweight.tol = 1e-8, reweight.max.iter = 10,
  trace_parms = FALSE, ...)
```

Arguments

mkinmod	A list of class mkinmod, containing the kinetic model to be fitted to the data, or one of the shorthand names ("SFO", "FOMC", "DFOP", "HS", "SFORB"). If a shorthand name is given, a parent only degradation model is generated for the variable with the highest value in observed.
observed	The observed data. It has to be in the long format as described in modFit, i.e. the first column called "name" must contain the name of the observed variable for each data point. The second column must contain the times of observation, named "time". The third column must be named "value" and contain the observed values. Optionally, a further column can contain weights for each data point. Its name must be passed as a further argument named err which is then passed on to modFit.
parms.ini	A named vector of initial values for the parameters, including parameters to be optimised and potentially also fixed parameters as indicated by fixed_parms. If set to "auto", initial values for rate constants are set to default values. Using parameter names that are not in the model gives an error. It is possible to only specify a subset of the parameters that the model needs. You can use the parameter lists "bparms.ode" from a previously fitted model, which contains the differential equation parameters from this model. This works nicely if the models are nested. An example is given below.
state.ini	A named vector of initial values for the state variables of the model. In case the observed variables are represented by more than one model variable, the names will differ from the names of the observed variables (see map component of mkinmod). The default is to set the initial value of the first model variable to the mean of the time zero values for the variable with the maximum observed value, and all others to 0. If this variable has no time zero observations, its initial value is set to 100.
fixed_parms	The names of parameters that should not be optimised but rather kept at the values specified in parms.ini.
fixed_initials	The names of model variables for which the initial state at time 0 should be excluded from the optimisation. Defaults to all state variables except for the first one.
from_max_mean	If this is set to TRUE, and the model has only one observed variable, then data before the time of the maximum observed value (after averaging for each sampling time) are discarded, and this time is subtracted from all remaining time values, so the time of the maximum observed mean value is the new time zero.
solution_type	If set to "eigen", the solution of the system of differential equations is based on the spectral decomposition of the coefficient matrix in cases that this is possible. If set to "deSolve", a numerical ode solver from package deSolve is used. If set to "analytical", an analytical solution of the model is used. This is only implemented for simple degradation experiments with only one state variable, i.e. with no metabolites. The default is "auto", which uses "analytical" if possible, otherwise "eigen" if the model can be expressed using eigenvalues and eigenvectors, and finally "deSolve" for the remaining models (time dependence of degradation rates and metabolites). This argument is passed on to the helper function mkinpredict.

`method.ode` The solution method passed via `mkinpredict` to `ode` in case the solution type is "deSolve". The default "lsoda" is performant, but sometimes fails to converge.

`use_compiled` If set to `FALSE`, no compiled version of the `mkinmod` model is used, in the calls to `mkinpredict` even if a compiled verion is present.

`method.modFit` The optimisation method passed to `modFit`.

In order to optimally deal with problems where local minima occur, the "Port" algorithm is now used per default as it is less prone to get trapped in local minima and depends less on starting values for parameters than the Levenberg Marquardt variant selected by "Marq". However, "Port" needs more iterations.

The former default "Marq" is the Levenberg Marquardt algorithm `nls.lm` from the package `minpack.lm` and usually needs the least number of iterations.

The "Pseudo" algorithm is not included because it needs finite parameter bounds which are currently not supported.

The "Newton" algorithm is not included because its number of iterations can not be controlled by `control.modFit` and it does not appear to provide advantages over the other algorithms.

`maxit.modFit` Maximum number of iterations in the optimisation. If not "auto", this will be passed to the method called by `modFit`, overriding what may be specified in the next argument `control.modFit`.

`control.modFit` Additional arguments passed to the optimisation method used by `modFit`.

`transform_rates`

Boolean specifying if kinetic rate constants should be transformed in the model specification used in the fitting for better compliance with the assumption of normal distribution of the estimator. If TRUE, also alpha and beta parameters of the FOMC model are log-transformed, as well as k1 and k2 rate constants for the DFOP and HS models and the break point tb of the HS model. If FALSE, zero is used as a lower bound for the rates in the optimisation.

`transform_fractions`

Boolean specifying if formation fractions constants should be transformed in the model specification used in the fitting for better compliance with the assumption of normal distribution of the estimator. The default (TRUE) is to do transformations. If TRUE, the g parameter of the DFOP and HS models are also transformed, as they can also be seen as compositional data. The transformation used for these transformations is the `ilr` transformation.

`plot` Should the observed values and the numerical solutions be plotted at each stage of the optimisation?

`quiet` Suppress printing out the current model cost after each 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.

`weight` only if `err=NULL`: how to weight 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.

`atol` Absolute error tolerance, passed to `ode`. Default is 1e-8, lower than in `lsoda`.

rtol — Absolute error tolerance, passed to ode. Default is 1e-10, much lower than in lsoda.

n.outtimes — The length of the dataseries that is produced by the model prediction function mkinpredict. This impacts the accuracy of the numerical solver if that is used (see solution_type argument. The default value is 100.

reweight.method — The method used for iteratively reweighting residuals, also known as iteratively reweighted least squares (IRLS). Default is NULL, the other method implemented is called "obs", meaning that each observed variable is assumed to have its own variance, this is estimated from the fit and used for weighting the residuals in each iteration until convergence of this estimate up to reweight.tol or up to the maximum number of iterations specified by reweight.max.iter.

reweight.tol — Tolerance for convergence criterion for the variance components in IRLS fits.

reweight.max.iter — Maximum iterations in IRLS fits.

trace_parms — Should a trace of the parameter values be listed?

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

Value

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

Note

The implementation of iteratively reweighted least squares is inspired by the work of the KinGUII team at Bayer Crop Science (Walter Schmitt and Zhenglei Gao). A similar implemention can also be found in CAKE 2.0, which is the other GUI derivative of mkin, sponsored by Syngenta.

Note

When using the "IORE" submodel for metabolites, fitting with "transform_rates = TRUE" (the default) often leads to failures of the numerical ODE solver. In this situation it may help to switch off the internal rate transformation.

Author(s)

Johannes Ranke

See Also

Plotting methods plot.mkinfit and mkinparplot.

Fitting of several models to several datasets in a single call to mmkin.

Examples

```
# Use shorthand notation for parent only degradation
fit <- mkinfit("FOMC", FOCUS_2006_C, quiet = TRUE)
summary(fit)

# One parent compound, one metabolite, both single first order.
# Use mkinsub for convenience in model formulation. Pathway to sink included per default.
SFO_SFO <- mkinmod(
  parent = mkinsub("SFO", "m1"),
  m1 = mkinsub("SFO"))
# Fit the model to the FOCUS example dataset D using defaults
print(system.time(fit <- mkinfit(SFO_SFO, FOCUS_2006_D,
                           solution_type = "eigen", quiet = TRUE)))
coef(fit)
endpoints(fit)
## Not run:
# deSolve is slower when no C compiler (gcc) was available during model generation
print(system.time(fit.deSolve <- mkinfit(SFO_SFO, FOCUS_2006_D,
                           solution_type = "deSolve")))
coef(fit.deSolve)
endpoints(fit.deSolve)

## End(Not run)

# Use stepwise fitting, using optimised parameters from parent only fit, FOMC
## Not run:
FOMC_SFO <- mkinmod(
  parent = mkinsub("FOMC", "m1"),
  m1 = mkinsub("SFO"))
# Fit the model to the FOCUS example dataset D using defaults
fit.FOMC_SFO <- mkinfit(FOMC_SFO, FOCUS_2006_D, quiet = TRUE)
# Use starting parameters from parent only FOMC fit
fit.FOMC = mkinfit("FOMC", FOCUS_2006_D, quiet = TRUE)
fit.FOMC_SFO <- mkinfit(FOMC_SFO, FOCUS_2006_D, quiet = TRUE,
  parms.ini = fit.FOMC$bparms.ode)

# Use stepwise fitting, using optimised parameters from parent only fit, SFORB
SFORB_SFO <- mkinmod(
  parent = list(type = "SFORB", to = "m1", sink = TRUE),
  m1 = list(type = "SFO"))
# Fit the model to the FOCUS example dataset D using defaults
fit.SFORB_SFO <- mkinfit(SFORB_SFO, FOCUS_2006_D, quiet = TRUE)
fit.SFORB_SFO.deSolve <- mkinfit(SFORB_SFO, FOCUS_2006_D, solution_type = "deSolve",
                                 quiet = TRUE)
# Use starting parameters from parent only SFORB fit (not really needed in this case)
fit.SFORB = mkinfit("SFORB", FOCUS_2006_D, quiet = TRUE)
fit.SFORB_SFO <- mkinfit(SFORB_SFO, FOCUS_2006_D, parms.ini = fit.SFORB$bparms.ode, quiet = TRUE)

## End(Not run)

## Not run:
# Weighted fits, including IRLS
```

```
SFO_SFO.ff <- mkinmod(parent = mkinsub("SFO", "m1"),
                      m1 = mkinsub("SFO"), use_of_ff = "max")
f.noweight <- mkinfit(SFO_SFO.ff, FOCUS_2006_D, quiet = TRUE)
summary(f.noweight)
f.irls <- mkinfit(SFO_SFO.ff, FOCUS_2006_D, reweight.method = "obs", quiet = TRUE)
summary(f.irls)
f.w.mean <- mkinfit(SFO_SFO.ff, FOCUS_2006_D, weight = "mean", quiet = TRUE)
summary(f.w.mean)
f.w.value <- mkinfit(SFO_SFO.ff, subset(FOCUS_2006_D, value != 0), err = "value",
                     quiet = TRUE)
summary(f.w.value)

## End(Not run)

## Not run:
# Manual weighting
dw <- FOCUS_2006_D
errors <- c(parent = 2, m1 = 1)
dw$err.man <- errors[FOCUS_2006_D$name]
f.w.man <- mkinfit(SFO_SFO.ff, dw, err = "err.man", quiet = TRUE)
summary(f.w.man)
f.w.man.irls <- mkinfit(SFO_SFO.ff, dw, err = "err.man", quiet = TRUE,
                        reweight.method = "obs")
summary(f.w.man.irls)

## End(Not run)
```

`mkinmod` *Function to set up a kinetic model with one or more state variables*

Description

The function usually takes several expressions, each assigning a compound name to a list, specifying the kinetic model type and reaction or transfer to other observed compartments. Instead of specifying several expressions, a list of lists can be given in the speclist argument.

For the definition of model types and their parameters, the equations given in the FOCUS and NAFTA guidance documents are used.

Usage

```
mkinmod(..., use_of_ff = "min", speclist = NULL, quiet = FALSE, verbose = FALSE)
```

Arguments

`...`	For each observed variable, a list has to be specified as an argument, containing at least a component `type`, specifying the type of kinetics to use for the variable. Currently, single first order kinetics "SFO", indeterminate order rate equation kinetics "IORE", or single first order with reversible binding "SFORB"

are implemented for all variables, while "FOMC", "DFOP" and "HS" can additionally be chosen for the first variable which is assumed to be the source compartment. Additionally, each component of the list can include a character vector `to`, specifying names of variables to which a transfer is to be assumed in the model. If the argument `use_of_ff` is set to "min" (default) and the model for the compartment is "SFO" or "SFORB", an additional component of the list can be "sink=FALSE" effectively fixing the flux to sink to zero.

`use_of_ff` Specification of the use of formation fractions in the model equations and, if applicable, the coefficient matrix. If "min", a minimum use of formation fractions is made in order to avoid fitting the product of formation fractions and rate constants. If "max", formation fractions are always used.

`speclist` The specification of the observed variables and their submodel types and pathways can be given as a single list using this argument. Default is NULL.

`quiet` Should messages be suppressed?

`verbose` If `TRUE`, passed to `cfunction` if applicable to give detailed information about the C function being built.

Value

A list of class `mkinmod` for use with `mkinfit`, containing

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

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

`map` A list containing named character vectors for each observed variable, specifying the modelling variables by which it is represented.

`use_of_ff` The content of `use_of_ff` is passed on in this list component.

`coefmat` The coefficient matrix, if the system of differential equations can be represented by one.

Note

The IORE submodel is not well tested (yet). When using this model for metabolites, you may want to read the second note in the help page to `mkinfit`.

Author(s)

Johannes Ranke

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

NAFTA Technical Working Group on Pesticides (not dated) Guidance for Evaluating and Calculating Degradation Kinetics in Environmental Media

Examples

```
# Specify the SFO model (this is not needed any more, as we can now mkinfit("SFO", ...)
SFO <- mkinmod(parent = list(type = "SFO"))

# One parent compound, one metabolite, both single first order
SFO_SFO <- mkinmod(
  parent = mkinsub("SFO", "m1"),
  m1 = mkinsub("SFO"))

## Not run:
# The above model used to be specified like this, before the advent of mkinsub()
SFO_SFO <- mkinmod(
  parent = list(type = "SFO", to = "m1"),
  m1 = list(type = "SFO"))

# Show details of creating the C function
SFO_SFO <- mkinmod(
  parent = mkinsub("SFO", "m1"),
  m1 = mkinsub("SFO"), verbose = TRUE)

# If we have several parallel metabolites
# (compare tests/testthat/test_synthetic_data_for_UBA_2014.R)
m_synth_DFOP_par <- mkinmod(parent = mkinsub("DFOP", c("M1", "M2")),
                            M1 = mkinsub("SFO"),
                            M2 = mkinsub("SFO"),
                            use_of_ff = "max", quiet = TRUE)

fit_DFOP_par_c <- mkinfit(m_synth_DFOP_par,
                          synthetic_data_for_UBA_2014[[12]]$data,
                          quiet = TRUE)

## End(Not run)
```

`mkinparplot` *Function to plot the confidence intervals obtained using mkinfit*

Description

This function plots the confidence intervals for the parameters fitted using `mkinfit`.

Usage

```
mkinparplot(object)
```

Arguments

`object`	A fit represented in an `mkinfit` object.

Value

Nothing is returned by this function, as it is called for its side effect, namely to produce a plot.

Author(s)

Johannes Ranke

Examples

```
model <- mkinmod(
  T245 = mkinsub("SFO", to = c("phenol"), sink = FALSE),
  phenol = mkinsub("SFO", to = c("anisole")),
  anisole = mkinsub("SFO"), use_of_ff = "max")
fit <- mkinfit(model, subset(mccall81_245T, soil == "Commerce"), quiet = TRUE)
mkinparplot(fit)
```

---

mkinplot *Plot the observed data and the fitted model of an mkinfit object*

Description

Deprecated function. It now only calls the plot method `plot.mkinfit`.

Usage

```
mkinplot(fit, ...)
```

Arguments

`fit`	an object of class `mkinfit`.
`...`	further arguments passed to `plot.mkinfit`.

Value

The function is called for its side effect.

Author(s)

Johannes Ranke

`mkinpredict` *Produce predictions from a kinetic model using specific parameters*

Description

This function produces a time series for all the observed variables in a kinetic model as specified by `mkinmod`, using a specific set of kinetic parameters and initial values for the state variables.

Usage

```
mkinpredict(mkinmod, odeparms, odeini, outtimes, solution_type = "deSolve",
    use_compiled = "auto", method.ode = "lsoda", atol = 1e-08, rtol = 1e-10,
      map_output = TRUE, ...)
```

Arguments

`mkinmod`	A kinetic model as produced by `mkinmod`.
`odeparms`	A numeric vector specifying the parameters used in the kinetic model, which is generally defined as a set of ordinary differential equations.
`odeini`	A numeric vectory containing the initial values of the state variables of the model. Note that the state variables can differ from the observed variables, for example in the case of the SFORB model.
`outtimes`	A numeric vector specifying the time points for which model predictions should be generated.
`solution_type`	The method that should be used for producing the predictions. This should generally be "analytical" if there is only one observed variable, and usually "deSolve" in the case of several observed variables. The third possibility "eigen" is faster but not applicable to some models e.g. using FOMC for the parent compound.
`method.ode`	The solution method passed via `mkinpredict` to `ode` in case the solution type is "deSolve". The default "lsoda" is performant, but sometimes fails to converge.
`use_compiled`	If set to `FALSE`, no compiled version of the `mkinmod` model is used, even if is present.
`atol`	Absolute error tolerance, passed to `ode`. Default is 1e-8, lower than in `lsoda`.
`rtol`	Absolute error tolerance, passed to `ode`. Default is 1e-10, much lower than in `lsoda`.
`map_output`	Boolean to specify if the output should list values for the observed variables (default) or for all state variables (if set to FALSE).
`...`	Further arguments passed to the ode solver in case such a solver is used.

Value

A matrix in the same format as the output of `ode`.

Author(s)

Johannes Ranke

Examples

```
SFO <- mkinmod(degradinol = list(type = "SFO"))
# Compare solution types
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    solution_type = "analytical")
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    solution_type = "deSolve")
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    solution_type = "deSolve", use_compiled = FALSE)
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    solution_type = "eigen")


# Compare integration methods to analytical solution
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    solution_type = "analytical")[21,]
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    method = "lsoda")[21,]
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    method = "ode45")[21,]
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100), 0:20,
    method = "rk4")[21,]
# rk4 is not as precise here

# The number of output times used to make a lot of difference until the
# default for atol was adjusted
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100),
    seq(0, 20, by = 0.1))[201,]
mkinpredict(SFO, c(k_degradinol_sink = 0.3), c(degradinol = 100),
    seq(0, 20, by = 0.01))[2001,]

# Check compiled model versions - they are faster than the eigenvalue based solutions!
SFO_SFO = mkinmod(parent = list(type = "SFO", to = "m1"),
                  m1 = list(type = "SFO"))
system.time(
 print(mkinpredict(SFO_SFO, c(k_parent_m1 = 0.05, k_parent_sink = 0.1, k_m1_sink = 0.01),
              c(parent = 100, m1 = 0), seq(0, 20, by = 0.1),
              solution_type = "eigen")[201,]))
system.time(
 print(mkinpredict(SFO_SFO, c(k_parent_m1 = 0.05, k_parent_sink = 0.1, k_m1_sink = 0.01),
              c(parent = 100, m1 = 0), seq(0, 20, by = 0.1),
              solution_type = "deSolve")[201,]))
system.time(
 print(mkinpredict(SFO_SFO, c(k_parent_m1 = 0.05, k_parent_sink = 0.1, k_m1_sink = 0.01),
              c(parent = 100, m1 = 0), seq(0, 20, by = 0.1),
              solution_type = "deSolve", use_compiled = FALSE)[201,]))
```

`mkinresplot` *Function to plot residuals stored in an mkin object*

Description

This function plots the residuals for the specified subset of the observed variables from an mkinfit object. A combined plot of the fitted model and the residuals can be obtained using `plot.mkinfit` using the argument `show_residuals = TRUE`.

Usage

```
mkinresplot(object,
  obs_vars = names(object$mkinmod$map),
  xlim = c(0, 1.1 * max(object$data$time)),
  xlab = "Time", ylab = "Residual",
  maxabs = "auto", legend = TRUE, lpos = "topright", ...)
```

Arguments

`object`	A fit represented in an `mkinfit` object.
`obs_vars`	A character vector of names of the observed variables for which residuals should be plotted. Defaults to all observed variables in the model
`xlim`	plot range in x direction.
`xlab`	Label for the x axis. Defaults to "Time [days]".
`ylab`	Label for the y axis. Defaults to "Residual [% of applied radioactivity]".
`maxabs`	Maximum absolute value of the residuals. This is used for the scaling of the y axis and defaults to "auto".
`legend`	Should a legend be plotted? Defaults to "TRUE".
`lpos`	Where should the legend be placed? Default is "topright". Will be passed on to `legend`.
`...`	further arguments passed to `plot`.

Value

Nothing is returned by this function, as it is called for its side effect, namely to produce a plot.

Author(s)

Johannes Ranke

See Also

`mkinplot`, for a way to plot the data and the fitted lines of the mkinfit object.

Examples

```
model <- mkinmod(parent = mkinsub("SFO", "m1"), m1 = mkinsub("SFO"))
fit <- mkinfit(model, FOCUS_2006_D, quiet = TRUE)
mkinresplot(fit, "m1")
```

`mkinsub` *Function to set up a kinetic submodel for one state variable*

Description

This is a convenience function to set up the lists used as arguments for `mkinmod`.

Usage

```
mkinsub(submodel, to = NULL, sink = TRUE, full_name = NA)
```

Arguments

`submodel`	Character vector of length one to specify the submodel type. See `mkinmod` for the list of allowed submodel names.
`to`	Vector of the names of the state variable to which a transformation shall be included in the model.
`sink`	Should a pathway to sink be included in the model in addition to the pathways to other state variables?
`full_name`	An optional name to be used e.g. for plotting fits performed with the model. You can use non-ASCII characters here, but then your R code will not be portable, *i.e.* may produce unintended plot results on other operating systems or system configurations.

Value

A list for use with `mkinmod`.

Author(s)

Johannes Ranke

Examples

```
# One parent compound, one metabolite, both single first order.
SFO_SFO <- mkinmod(
  parent = list(type = "SFO", to = "m1"),
  m1 = list(type = "SFO"))

# The same model using mkinsub
SFO_SFO.2 <- mkinmod(
  parent = mkinsub("SFO", "m1"),
```

```
  m1 = mkinsub("SFO"))

# Now supplying full names
SFO_SFO.2 <- mkinmod(
  parent = mkinsub("SFO", "m1", full_name = "Test compound"),
  m1 = mkinsub("SFO", full_name = "Metabolite M1"))
```

`mkin_long_to_wide` *Convert a dataframe from long to wide format*

Description

This function takes a dataframe in the long form as required by `modCost` and converts it into a dataframe with one independent variable and several dependent variables as columns.

Usage

```
mkin_long_to_wide(long_data, time = "time", outtime = "time")
```

Arguments

`long_data`	The dataframe must contain one variable called "time" with the time values specified by the `time` argument, one column called "name" with the grouping of the observed values, and finally one column of observed values called "value".
`time`	The name of the time variable in the long input data.
`outtime`	The name of the time variable in the wide output data.

Value

Dataframe in wide format.

Author(s)

Johannes Ranke

Examples

```
mkin_long_to_wide(FOCUS_2006_D)
```

mkin_wide_to_long *Convert a dataframe with observations over time into long format*

Description

This function simply takes a dataframe with one independent variable and several dependent variable and converts it into the long form as required by `modCost`.

Usage

```
mkin_wide_to_long(wide_data, time = "t")
```

Arguments

`wide_data`	The dataframe must contain one variable with the time values specified by the `time` argument and usually more than one column of observed values.
`time`	The name of the time variable.

Value

Dataframe in long format as needed for `modCost`.

Author(s)

Johannes Ranke

Examples

```
wide <- data.frame(t = c(1,2,3), x = c(1,4,7), y = c(3,4,5))
mkin_wide_to_long(wide)
```

mmkin *Fit one or more kinetic models with one or more state variables to one or more datasets*

Description

This function calls `mkinfit` on all combinations of models and datasets specified in its first two arguments.

Usage

```
mmkin(models, datasets,
      cores = round(detectCores()/2), cluster = NULL, ...)
```

Arguments

models	Either a character vector of shorthand names ("SFO", "FOMC", "DFOP", "HS", "SFORB"), or an optionally named list of mkinmod objects.
datasets	An optionally named list of datasets suitable as observed data for mkinfit.
cores	The number of cores to be used for multicore processing. This is only used when the cluster argument is NULL.
cluster	A cluster as returned by makeCluster to be used for parallel execution.
...	Further arguments that will be passed to mkinfit.

Value

A matrix of mkinfit objects that can be indexed using the model and dataset names as row and column indices.

Author(s)

Johannes Ranke

See Also

[.mmkin for subsetting, plot.mmkin for plotting.

Examples

```
## Not run:
m_synth_SFO_lin <- mkinmod(parent = mkinsub("SFO", "M1"),
                           M1 = mkinsub("SFO", "M2"),
                           M2 = mkinsub("SFO"), use_of_ff = "max")

m_synth_FOMC_lin <- mkinmod(parent = mkinsub("FOMC", "M1"),
                            M1 = mkinsub("SFO", "M2"),
                            M2 = mkinsub("SFO"), use_of_ff = "max")

models <- list(SFO_lin = m_synth_SFO_lin, FOMC_lin = m_synth_FOMC_lin)
datasets <- lapply(synthetic_data_for_UBA_2014[1:3], function(x) x$data)
names(datasets) <- paste("Dataset", 1:3)

time_default <- system.time(fits.0 <- mmkin(models, datasets, quiet = TRUE))
time_1 <- system.time(fits.4 <- mmkin(models, datasets, cores = 1, quiet = TRUE))

time_default
time_1

endpoints(fits.0[["SFO_lin", 2]])

# plot.mkinfit handles rows or columns of mmkin result objects
plot(fits.0[1, ])
plot(fits.0[1, ], obs_var = c("M1", "M2"))
plot(fits.0[, 1])
# Use double brackets to extract a single mkinfit object, which will be plotted
```

```
# by plot.mkinfit and can be plotted using plot_sep
plot(fits.0[[1, 1]], sep_obs = TRUE, show_residuals = TRUE, show_errmin = TRUE)
plot_sep(fits.0[[1, 1]])
# Plotting with mmkin (single brackets, extracting an mmkin object) does not
# allow to plot the observed variables separately
plot(fits.0[1, 1])

## End(Not run)
```

`plot.mkinfit` *Plot the observed data and the fitted model of an mkinfit object*

Description

Solves the differential equations with the optimised and fixed parameters from a previous successful call to `mkinfit` and plots the observed data together with the solution of the fitted model.

If the current plot device is a `tikz` device, then latex is being used for the formatting of the chi2 error level, if `show_errmin = TRUE`.

Usage

```
## S3 method for class 'mkinfit'
plot(x, fit = x,
  obs_vars = names(fit$mkinmod$map),
  xlab = "Time", ylab = "Observed",
  xlim = range(fit$data$time),
  ylim = "default",
  col_obs = 1:length(obs_vars), pch_obs = col_obs,
  lty_obs = rep(1, length(obs_vars)),
  add = FALSE, legend = !add,
  show_residuals = FALSE, maxabs = "auto",
  sep_obs = FALSE, rel.height.middle = 0.9,
  lpos = "topright", inset = c(0.05, 0.05),
  show_errmin = FALSE, errmin_digits = 3, ...)
plot_sep(fit, sep_obs = TRUE,  show_residuals = TRUE, show_errmin = TRUE, ...)
```

Arguments

`x`	Alias for fit introduced for compatibility with the generic S3 method.
`fit`	An object of class `mkinfit`.
`obs_vars`	A character vector of names of the observed variables for which the data and the model should be plotted. Defauls to all observed variables in the model.
`xlab`	Label for the x axis.
`ylab`	Label for the y axis.
`xlim`	Plot range in x direction.
`ylim`	Plot range in y direction.

col_obs	Colors used for plotting the observed data and the corresponding model prediction lines.
pch_obs	Symbols to be used for plotting the data.
lty_obs	Line types to be used for the model predictions.
add	Should the plot be added to an existing plot?
legend	Should a legend be added to the plot?
show_residuals	Should residuals be shown? If only one plot of the fits is shown, the residual plot is in the lower third of the plot? Otherwise, i.e. if "sep_obs" is given, the residual plots will be located to the right of the plots of the fitted curves.
maxabs	Maximum absolute value of the residuals. This is used for the scaling of the y axis and defaults to "auto".
sep_obs	Should the observed variables be shown in separate subplots? If yes, residual plots requested by "show_residuals" will be shown next to, not below the plot of the fits.
rel.height.middle	The relative height of the middle plot, if more than two rows of plots are shown.
lpos	Position(s) of the legend(s). Passed to legend as the first argument. If not length one, this should be of the same length as the obs_var argument.
inset	Passed to legend if applicable.
show_errmin	Should the FOCUS chi2 error value be shown in the upper margin of the plot?
errmin_digits	The number of significant digits for rounding the FOCUS chi2 error percentage.
...	Further arguments passed to plot.

Value

The function is called for its side effect.

Author(s)

Johannes Ranke

Examples

```
# One parent compound, one metabolite, both single first order, path from
# parent to sink included, use Levenberg-Marquardt for speed
SFO_SFO <- mkinmod(parent = mkinsub("SFO", "m1", full = "Parent"),
                   m1 = mkinsub("SFO", full = "Metabolite M1" ))
fit <- mkinfit(SFO_SFO, FOCUS_2006_D, quiet = TRUE, method.modFit = "Marq")
plot(fit)
plot(fit, show_residuals = TRUE)

# Show the observed variables separately
plot(fit, sep_obs = TRUE, lpos = c("topright", "bottomright"))

# Show the observed variables separately, with residuals
plot(fit, sep_obs = TRUE, show_residuals = TRUE, lpos = c("topright", "bottomright"),
     show_errmin = TRUE)
```

```
# The same can be obtained with less typing, using the convenience function plot_sep
plot_sep(fit, lpos = c("topright", "bottomright"))
```

`plot.mmkin` *Plot model fits (observed and fitted) and the residuals for a row or column of an mmkin object*

Description

When x is a row selected from an mmkin object (`[.mmkin`), the same model fitted for at least one dataset is shown. When it is a column, the fit of at least one model to the same dataset is shown.

If the current plot device is a `tikz` device, then latex is being used for the formatting of the chi2 error level.

Usage

```
## S3 method for class 'mmkin'
plot(x, main = "auto", legends = 1, errmin_var = "All data", errmin_digits = 3,
             cex = 0.7, rel.height.middle = 0.9, ...)
```

Arguments

`x`	An object of class `mmkin`, with either one row or one column.
`main`	The main title placed on the outer margin of the plot.
`legends`	An index for the fits for which legends should be shown.
`errmin_var`	The variable for which the FOCUS chi2 error value should be shown.
`errmin_digits`	The number of significant digits for rounding the FOCUS chi2 error percentage.
`cex`	Passed to the plot functions and `mtext`.
`rel.height.middle`	The relative height of the middle plot, if more than two rows of plots are shown.
`...`	Further arguments passed to `plot.mkinfit` and `mkinresplot`.

Value

The function is called for its side effect.

Author(s)

Johannes Ranke

Examples

```
# Only use one core not to offend CRAN checks, use Levenberg-Marquardt for speed
fits <- mmkin(c("FOMC", "HS"),
          list("FOCUS B" = FOCUS_2006_B, "FOCUS C" = FOCUS_2006_C), # named list for titles
             cores = 1, quiet = TRUE, method.modFit = "Marq")
plot(fits[, "FOCUS C"])
plot(fits["FOMC", ])

# We can also plot a single fit, if we like the way plot.mmkin works, but then the plot
# height should be smaller than the plot width (this is not possible for the html pages
# generated by pkgdown, as far as I know).
plot(fits["FOMC", "FOCUS C"]) # same as plot(fits[1, 2])
```

`print.mkinds` *Print mkinds objects*

Description

Print mkinds objects.

Usage

```
## S3 method for class 'mkinds'
print(x, ...)
```

Arguments

`x`	An `mkinds` object.
`...`	Not used.

`print.mkinmod` *Print mkinmod objects*

Description

Print mkinmod objects in a way that the user finds his way to get to its components.

Usage

```
## S3 method for class 'mkinmod'
print(x, ...)
```

Arguments

`x`	An `mkinmod` object.
`...`	Not used.

Examples

```
m_synth_SFO_lin <- mkinmod(parent = list(type = "SFO", to = "M1"),
                           M1 = list(type = "SFO", to = "M2"),
                           M2 = list(type = "SFO"), use_of_ff = "max")

print(m_synth_SFO_lin)
```

`schaefer07_complex_case` *Metabolism data set used for checking the software quality of KinGUI*

Description

This dataset was used for a comparison of KinGUI and ModelMaker to check the software quality of KinGUI in the original publication (Schäfer et al., 2007). The results from the fitting are also included.

Usage

```
data(schaefer07_complex_case)
```

Format

The data set is a data frame with 8 observations on the following 6 variables.

`time` a numeric vector

`parent` a numeric vector

`A1` a numeric vector

`B1` a numeric vector

`C1` a numeric vector

`A2` a numeric vector

The results are a data frame with 14 results for different parameter values

References

Schäfer D, Mikolasch B, Rainbird P and Harvey B (2007). KinGUI: a new kinetic software tool for evaluations according to FOCUS degradation kinetics. In: Del Re AAM, Capri E, Fragoulis G and Trevisan M (Eds.). Proceedings of the XIII Symposium Pesticide Chemistry, Piacenza, 2007, p. 916-923.

Examples

```
data <- mkin_wide_to_long(schaefer07_complex_case, time = "time")
model <- mkinmod(
  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"), use_of_ff = "max")
  ## Not run:
    fit <- mkinfit(model, data, quiet = TRUE)
    plot(fit)
    endpoints(fit)

## End(Not run)
 # Compare with the results obtained in the original publication
 print(schaefer07_complex_results)
```

`SFO.solution` *Single First-Order kinetics*

Description

Function describing exponential decline from a defined starting value.

Usage

```
SFO.solution(t, parent.0, k)
```

Arguments

`t`	Time.
`parent.0`	Starting value for the response variable at time zero.
`k`	Kinetic constant.

Value

The value of the response variable at time `t`.

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
## Not run: plot(function(x) SFO.solution(x, 100, 3), 0, 2)
```

SFORB.solution *Single First-Order Reversible Binding kinetics*

Description

Function describing the solution of the differential equations describing the kinetic model with first-order terms for a two-way transfer from a free to a bound fraction, and a first-order degradation term for the free fraction. The initial condition is a defined amount in the free fraction and no substance in the bound fraction.

Usage

```
SFORB.solution(t, parent.0, k_12, k_21, k_1output)
```

Arguments

t	Time.
parent.0	Starting value for the response variable at time zero.
k_12	Kinetic constant describing transfer from free to bound.
k_21	Kinetic constant describing transfer from bound to free.
k_1output	Kinetic constant describing degradation of the free fraction.

Value

The value of the response variable, which is the sum of free and bound fractions at time t.

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
## Not run: plot(function(x) SFORB.solution(x, 100, 0.5, 2, 3), 0, 2)
```

summary.mkinfit *Summary method for class "mkinfit"*

Description

Lists model equations, the summary as returned by summary.modFit, the chi2 error levels calculated according to FOCUS guidance (2006) as far as defined therein, and optionally the data, consisting of observed, predicted and residual values.

Usage

```
## S3 method for class 'mkinfit'
summary(object, data = TRUE, distimes = TRUE, alpha = 0.05, ...)
## S3 method for class 'summary.mkinfit'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	an object of class mkinfit.
x	an object of class summary.mkinfit.
data	logical, indicating whether the data should be included in the summary.
distimes	logical, indicating whether DT50 and DT90 values should be included.
alpha	error level for confidence interval estimation from t distribution
digits	Number of digits to use for printing
...	optional arguments passed to methods like print.

Value

The summary function returns a list derived from summary.modFit, with components, among others

version, Rversion
: The mkin and R versions used

date.fit, date.summary
: The dates where the fit and the summary were produced

use_of_ff
: Was maximum or minimum use made of formation fractions

residuals, residualVariance, sigma, modVariance, df
: As in summary.modFit

cov.unscaled, cov.scaled, info, niter, stopmess, par
: As in summary.modFit

bpar
: Optimised and backtransformed parameters

diffs
: The differential equations used in the model

data
: The data (see Description above).

start
: The starting values and bounds, if applicable, for optimised parameters.

fixed	The values of fixed parameters.
errmin	The chi2 error levels for each observed variable.
bparms.ode	All backtransformed ODE parameters, for use as starting parameters for related models.
ff	The estimated formation fractions derived from the fitted model.
distimes	The DT50 and DT90 values for each observed variable.
SFORB	If applicable, eigenvalues of SFORB components of the model.

The print method is called for its side effect, i.e. printing the summary.

Author(s)

Johannes Ranke

References

FOCUS (2006) "Guidance Document on Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, EC Document Reference Sanco/10058/2005 version 2.0, 434 pp, http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics

Examples

```
summary(mkinfit(mkinmod(parent = mkinsub("SFO")), FOCUS_2006_A, quiet = TRUE))
```

---

synthetic_data_for_UBA_2014

Synthetic datasets for one parent compound with two metabolites

---

Description

The 12 datasets were generated using four different models and three different variance components. The four models are either the SFO or the DFOP model with either two sequential or two parallel metabolites.

Variance component 'a' is based on a normal distribution with standard deviation of 3, Variance component 'b' is also based on a normal distribution, but with a standard deviation of 7. Variance component 'c' is based on the error model from Rocke and Lorenzato (1995), with the minimum standard deviation (for small y values) of 0.5, and a proportionality constant of 0.07 for the increase of the standard deviation with y.

Initial concentrations for metabolites and all values where adding the variance component resulted in a value below the assumed limit of detection of 0.1 were set to NA.

As an example, the first dataset has the title SFO_lin_a and is based on the SFO model with two sequential metabolites (linear pathway), with added variance component 'a'.

Compare also the code in the example section to see the degradation models.

Usage

```
synthetic_data_for_UBA_2014
```

Format

A list containing datasets in the form internally used by the 'gmkin' package. The list has twelfe components. Each of the components is one dataset that has, among others, the following components

`title` The name of the dataset, e.g. `SFO_lin_a`

`data` A data frame with the data in the form expected by `mkinfit`

Source

Ranke (2014) Prüfung und Validierung von Modellierungssoftware als Alternative zu ModelMaker 4.0, Umweltbundesamt Projektnummer 27452

Rocke, David M. und Lorenzato, Stefan (1995) A two-component model for measurement error in analytical chemistry. Technometrics 37(2), 176-184.

Examples

```
## Not run:
# The data have been generated using the following kinetic models
m_synth_SFO_lin <- mkinmod(parent = list(type = "SFO", to = "M1"),
                           M1 = list(type = "SFO", to = "M2"),
                           M2 = list(type = "SFO"), use_of_ff = "max")


m_synth_SFO_par <- mkinmod(parent = list(type = "SFO", to = c("M1", "M2"),
                                         sink = FALSE),
                           M1 = list(type = "SFO"),
                           M2 = list(type = "SFO"), use_of_ff = "max")

m_synth_DFOP_lin <- mkinmod(parent = list(type = "DFOP", to = "M1"),
                            M1 = list(type = "SFO", to = "M2"),
                            M2 = list(type = "SFO"), use_of_ff = "max")

m_synth_DFOP_par <- mkinmod(parent = list(type = "DFOP", to = c("M1", "M2"),
                                          sink = FALSE),
                            M1 = list(type = "SFO"),
                            M2 = list(type = "SFO"), use_of_ff = "max")

# The model predictions without intentional error were generated as follows
sampling_times = c(0, 1, 3, 7, 14, 28, 60, 90, 120)

d_synth_SFO_lin <- mkinpredict(m_synth_SFO_lin,
                               c(k_parent = 0.7, f_parent_to_M1 = 0.8,
                                 k_M1 = 0.3, f_M1_to_M2 = 0.7,
                                 k_M2 = 0.02),
                               c(parent = 100, M1 = 0, M2 = 0),
                               sampling_times)
```

```
d_synth_DFOP_lin <- mkinpredict(m_synth_DFOP_lin,
                                c(k1 = 0.2, k2 = 0.02, g = 0.5,
                                  f_parent_to_M1 = 0.5, k_M1 = 0.3,
                                  f_M1_to_M2 = 0.7, k_M2 = 0.02),
                                c(parent = 100, M1 = 0, M2 = 0),
                                sampling_times)

d_synth_SFO_par <- mkinpredict(m_synth_SFO_par,
                               c(k_parent = 0.2,
                                 f_parent_to_M1 = 0.8, k_M1 = 0.01,
                                 f_parent_to_M2 = 0.2, k_M2 = 0.02),
                                 c(parent = 100, M1 = 0, M2 = 0),
                                 sampling_times)

d_synth_DFOP_par <- mkinpredict(m_synth_DFOP_par,
                               c(k1 = 0.3, k2 = 0.02, g = 0.7,
                                 f_parent_to_M1 = 0.6, k_M1 = 0.04,
                                 f_parent_to_M2 = 0.4, k_M2 = 0.01),
                                 c(parent = 100, M1 = 0, M2 = 0),
                                 sampling_times)

# Construct names for datasets with errors
d_synth_names = paste0("d_synth_", c("SFO_lin", "SFO_par",
                                     "DFOP_lin", "DFOP_par"))

# Function for adding errors. The add_err function now published with this
# package is a slightly generalised version where the names of secondary
# compartments that should have an initial value of zero (M1 and M2 in this
# case) are not hardcoded any more.
add_err = function(d, sdfunc, LOD = 0.1, reps = 2, seed = 123456789)
{
  set.seed(seed)
  d_long = mkin_wide_to_long(d, time = "time")
  d_rep = data.frame(lapply(d_long, rep, each = 2))
  d_rep$value = rnorm(length(d_rep$value), d_rep$value, sdfunc(d_rep$value))

  d_rep[d_rep$time == 0 & d_rep$name
  d_NA <- transform(d_rep, value = ifelse(value < LOD, NA, value))
  d_NA$value <- round(d_NA$value, 1)
  return(d_NA)
}

# The following is the two-component model of Rocke and Lorenzato (1995)
sdfunc_twocomp = function(value, sd_low, rsd_high) {
  sqrt(sd_low^2 + value^2 * rsd_high^2)
}

# Add the errors.
for (d_synth_name in d_synth_names)
{
  d_synth = get(d_synth_name)
  assign(paste0(d_synth_name, "_a"), add_err(d_synth, function(value) 3))
```

```
  assign(paste0(d_synth_name, "_b"), add_err(d_synth, function(value) 7))
  assign(paste0(d_synth_name, "_c"), add_err(d_synth,
                          function(value) sdfunc_twocomp(value, 0.5, 0.07)))

}

d_synth_err_names = c(
  paste(rep(d_synth_names, each = 3), letters[1:3], sep = "_")
)

# This is just one example of an evaluation using the kinetic model used for
# the generation of the data
fit <- mkinfit(m_synth_SFO_lin, synthetic_data_for_UBA_2014[[1]]$data,
               quiet = TRUE)
plot_sep(fit)
summary(fit)

## End(Not run)
```

`transform_odeparms` *Functions to transform and backtransform kinetic parameters for fitting*

Description

The transformations are intended to map parameters that should only take on restricted values to the full scale of real numbers. For kinetic rate constants and other paramters that can only take on positive values, a simple log transformation is used. For compositional parameters, such as the formations fractions that should always sum up to 1 and can not be negative, the `ilr` transformation is used.

The transformation of sets of formation fractions is fragile, as it supposes the same ordering of the components in forward and backward transformation. This is no problem for the internal use in `mkinfit`.

Usage

```
transform_odeparms(parms, mkinmod,
                   transform_rates = TRUE, transform_fractions = TRUE)
backtransform_odeparms(transparms, mkinmod,
                       transform_rates = TRUE, transform_fractions = TRUE)
```

Arguments

`parms`	Parameters of kinetic models as used in the differential equations.
`transparms`	Transformed parameters of kinetic models as used in the fitting procedure.
`mkinmod`	The kinetic model of class `mkinmod`, containing the names of the model variables that are needed for grouping the formation fractions before `ilr` transformation, the parameter names and the information if the pathway to sink is included in the model.

`transform_rates`
: Boolean specifying if kinetic rate constants should be transformed in the model specification used in the fitting for better compliance with the assumption of normal distribution of the estimator. If TRUE, also alpha and beta parameters of the FOMC model are log-transformed, as well as k1 and k2 rate constants for the DFOP and HS models and the break point tb of the HS model.

`transform_fractions`
: Boolean specifying if formation fractions constants should be transformed in the model specification used in the fitting for better compliance with the assumption of normal distribution of the estimator. The default (TRUE) is to do transformations. The g parameter of the DFOP and HS models are also transformed, as they can also be seen as compositional data. The transformation used for these transformations is the `ilr` transformation.

Value

A vector of transformed or backtransformed parameters with the same names as the original parameters.

Author(s)

Johannes Ranke

Examples

```
SFO_SFO <- mkinmod(
  parent = list(type = "SFO", to = "m1", sink = TRUE),
  m1 = list(type = "SFO"))
# Fit the model to the FOCUS example dataset D using defaults
fit <- mkinfit(SFO_SFO, FOCUS_2006_D, quiet = TRUE)
summary(fit, data=FALSE) # See transformed and backtransformed parameters

## Not run:
fit.2 <- mkinfit(SFO_SFO, FOCUS_2006_D, transform_rates = FALSE, quiet = TRUE)
summary(fit.2, data=FALSE)

## End(Not run)

initials <- fit$start$value
names(initials) <- rownames(fit$start)
transformed <- fit$start_transformed$value
names(transformed) <- rownames(fit$start_transformed)
transform_odeparms(initials, SFO_SFO)
backtransform_odeparms(transformed, SFO_SFO)

## Not run:
# The case of formation fractions
SFO_SFO.ff <- mkinmod(
  parent = list(type = "SFO", to = "m1", sink = TRUE),
  m1 = list(type = "SFO"),
  use_of_ff = "max")
```

```
fit.ff <- mkinfit(SFO_SFO.ff, FOCUS_2006_D, quiet = TRUE)
summary(fit.ff, data = FALSE)
initials <- c("f_parent_to_m1" = 0.5)
transformed <- transform_odeparms(initials, SFO_SFO.ff)
backtransform_odeparms(transformed, SFO_SFO.ff)

# And without sink
SFO_SFO.ff.2 <- mkinmod(
  parent = list(type = "SFO", to = "m1", sink = FALSE),
  m1 = list(type = "SFO"),
  use_of_ff = "max")


fit.ff.2 <- mkinfit(SFO_SFO.ff.2, FOCUS_2006_D, quiet = TRUE)
summary(fit.ff.2, data = FALSE)

## End(Not run)
```

`[.mmkin` *Subsetting method for mmkin objects*

Description

Subsetting method for mmkin objects.

Usage

```
## S3 method for class 'mmkin'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	An `mmkin` object
`i`	Row index selecting the fits for specific models
`j`	Column index selecting the fits to specific datasets
`...`	Not used, only there to satisfy the generic method definition
`drop`	If FALSE, the method always returns an mmkin object, otherwise either a list of mkinfit objects or a single mkinfit object.

Value

An object of class `mmkin`.

Author(s)

Johannes Ranke

Examples

```
# Only use one core, to pass R CMD check --as-cran
fits <- mmkin(c("SFO", "FOMC"), list(B = FOCUS_2006_B, C = FOCUS_2006_C),
              cores = 1, quiet = TRUE)
fits["FOMC", ]
fits[, "B"]
fits["SFO", "B"]

head(
  # This extracts an mkinfit object with lots of components
  fits[["FOMC", "B"]]
)

head(
# The same can be achieved by
  fits["SFO", "B", drop = TRUE]
)
```

Index