

Package ‘mkin’

July 4, 2019

Type Package

Title Kinetic Evaluation of Chemical Degradation Data

Version 0.9.49.5

Date 2019-07-04

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, deSolve, R6, inline, parallel, plyr, numDeriv

Suggests knitr, rbenchmark, tikzDevice, testthat, rmarkdown, covr, vdiff, benchmarkme, tibble

License GPL

LazyLoad yes

LazyData yes

Encoding UTF-8

VignetteBuilder knitr

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

URL <https://pkgdown.jrwb.de/mkin>

RoxygenNote 6.1.1

NeedsCompilation no

Author Johannes Ranke [aut, cre, cph]
(<<https://orcid.org/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 2019-07-04 09:40:05 UTC

R topics documented:

add_err	3
AIC.mmkin	5
CAKE_export	6
DFOP.solution	7
endpoints	8
experimental_data_for_UBA_2019	8
FOCUS_2006_datasets	11
FOCUS_2006_DFOP_ref_A_to_B	11
FOCUS_2006_FOMC_ref_A_to_F	12
FOCUS_2006_HS_ref_A_to_F	13
FOCUS_2006_SFO_ref_A_to_F	14
FOMC.solution	15
geometric_mean	16
HS.solution	16
ilr	17
IORE.solution	18
logistic.solution	19
logLik.mkinfit	20
max_twa_parent	22
mccall81_245T	23
mkin	24
mkinerrmin	24
mkinerrplot	26
mkinfit	27
mkinmod	32
mkinparplot	34
mkinplot	35
mkinpredict	35
mkinresplot	37
mkin	39
mkin_long_to_wide	40
mkin_wide_to_long	40
mmkin	41
nafta	43
NAFTA_SOP_2015	44
NAFTA_SOP_Attachment	44
plot.mkinfit	45
plot.mmkin	47
plot.nafta	49
print.mkin	49
print.mkinmod	50

print.nafta	50
schaefer07_complex_case	51
SFO.solution	52
SFORB.solution	53
sigma_twocomp	54
summary.mkinfit	55
synthetic_data_for_UBA_2014	56
test_data_from_UBA_2014	59
transform_odeparms	60
[.mmkin	62

Index	64
--------------	-----------

add_err	<i>Add normally distributed errors to simulated kinetic degradation data</i>
---------	--

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_SF0_SF0 <- mkinmod(parent = mkinsub("SF0", "M1"),
                    M1 = mkinsub("SF0"), 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_SF0_SF0 <- mkinpredict(m_SF0_SF0,
                        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_SF0_SF0_err <- add_err(d_SF0_SF0, function(x) 10, n = 3, seed = 123456789 )

# Name the datasets for nicer plotting
names(d_SF0_SF0_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 only one core not to offend CRAN
# checks
## Not run:
f_SF0_SF0 <- mmkin(list("SF0-SF0" = m_SF0_SF0),
                  d_SF0_SF0_err, cores = 1,
                  quiet = TRUE)

plot(f_SF0_SF0)

# 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_SF0_SF0[[3]], show_residuals = TRUE)

# If we use single brackets, we should give two indices (model and dataset),
# and plot.mmkin is used
plot(f_SF0_SF0[1, 3])

## End(Not run)
```

`AIC.mmkin`*Calculated the AIC for a column of an mmkin object*

Description

Provides a convenient way to compare different kineti models fitted to the same dataset.

Usage

```
## S3 method for class 'mmkin'  
AIC(object, ..., k = 2)
```

Arguments

<code>object</code>	An object of class <code>mmkin</code> , containing only one column.
<code>...</code>	For compatibility with the generic method
<code>k</code>	As in the generic method

Value

As in the generic method (a numeric value for single fits, or a dataframe if there are several fits in the column).

Author(s)

Johannes Ranke

Examples

```
f <- mmkin(c("SFO", "FOMC", "DFOP"),  
  list("FOCUS A" = FOCUS_2006_A,  
        "FOCUS C" = FOCUS_2006_C), cores = 1, quiet = TRUE)  
AIC(f[1, "FOCUS A"]) # We get a single number for a single fit  
  
# For FOCUS A, the models fit almost equally well, so the higher the number  
# of parameters, the higher (worse) the AIC  
AIC(f[, "FOCUS A"])  
AIC(f[, "FOCUS A"], k = 0) # If we do not penalize additional parameters, we get nearly the same  
  
# For FOCUS C, the more complex models fit better  
AIC(f[, "FOCUS C"])
```

CAKE_export

*Export a list of datasets format to a CAKE study file***Description**

In addition to the datasets, the pathways in the degradation model can be specified as well.

Usage

```
CAKE_export(ds, map = c(parent = "Parent"),
  links = NA,
  filename = "CAKE_export.csf", path = ".", overwrite = FALSE,
  study = "Codlemone aerobic soil degradation",
  description = "",
  time_unit = "days",
  res_unit = "% AR",
  comment = "Created using mkin::CAKE_export",
  date = Sys.Date(),
  optimiser = "IRLS")
```

Arguments

ds	A named list of datasets in long format as compatible with mkinfit .
map	A character vector with CAKE compartment names (Parent, A1, ...), named with the names used in the list of datasets.
links	An optional character vector of target compartments, named with the names of the source compartments. In order to make this easier, the names are used as in the datasets supplied.
filename	Where to write the result. Should end in .csf in order to be compatible with CAKE.
path	An optional path to the output file.
overwrite	If TRUE, existing files are overwritten.
study	The name of the study.
description	An optional description.
time_unit	The time unit for the residue data.
res_unit	The unit used for the residues.
comment	An optional comment.
date	The date of file creation.
optimiser	Can be OLS or IRLS.

Value

The function is called for its side effect.

Author(s)

Johannes Ranke

`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	<i>Function to calculate endpoints for further use from kinetic models fitted with mkinfit</i>
-----------	--

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)
```


Description

The 12 datasets were extracted from active substance evaluation dossiers published by EFSA. Kinetic evaluations shown for these datasets are intended to illustrate and advance error model specifications. The fact that these data and some results are shown here do not imply a license to use them in the context of pesticide registrations, as the use of the data may be constrained by data protection regulations.

Preprocessing of data was performed based on the recommendations of the FOCUS kinetics workgroup (FOCUS, 2014) as described below.

Datasets 1 and 2 are from the Renewal Assessment Report (RAR) for imazamox (France, 2015, p. 15). For setting values reported as zero, an LOQ of 0.1 was assumed. Metabolite residues reported for day zero were added to the parent compound residues.

Datasets 3 and 4 are from the Renewal Assessment Report (RAR) for isofetamid (Belgium, 2014, p. 8) and show the data for two different radiolabels. For dataset 4, the value given for the metabolite in the day zero sampling in replicate B was added to the parent compound, following the respective FOCUS recommendation.

Dataset 5 is from the Renewal Assessment Report (RAR) for ethofumesate (Austria, 2015, p. 16).

Datasets 6 to 10 are from the Renewal Assessment Report (RAR) for glyphosate (Germany, 2013a, pages 8, 28, 50, 51). For the initial sampling, the residues given for the metabolite were added to the parent value, following the recommendation of the FOCUS kinetics workgroup.

Dataset 11 is from the Renewal Assessment Report (RAR) for 2,4-D (Germany, 2013b, p. 644). Values reported as zero were set to NA, with the exception of the day three sampling of metabolite A2, which was set to one half of the LOD reported to be 1% AR.

Dataset 12 is from the Renewal Assessment Report (RAR) for thifensulfuron-methyl (United Kingdom, 2014, p. 81).

Usage

```
experimental_data_for_UBA_2019
```

Format

A list containing twelve datasets as an R6 class defined by `mkind`s, each containing, among others, the following components

`title` The name of the dataset, e.g. Soil 1

`data` A data frame with the data in the form expected by `mkinf`t

Source

Austria (2015). Ethofumesate Renewal Assessment Report Volume 3 Annex B.8 (AS)

Belgium (2014). Isofetamid (IKF-5411) Draft Assessment Report Volume 3 Annex B.8 (AS)

France (2015). Imazamox Draft Renewal Assessment Report Volume 3 Annex B.8 (AS)

FOCUS (2014) "Generic guidance for Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration" Report of the FOCUS Work Group on Degradation Kinetics, Version 1.1, 18 December 2014 <http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics>

Germany (2013a). Renewal Assessment Report Glyphosate Volume 3 Annex B.8: Environmental Fate and Behaviour

Germany (2013b). Renewal Assessment Report 2,4-D Volume 3 Annex B.8: Fate and behaviour in the environment

Ranke (2019) Documentation of results obtained for the error model expertise written for the German Umweltbundesamt.

United Kingdom (2014). Thifensulfuron-methyl - Annex B.8 (Volume 3) to the Report and Proposed Decision of the United Kingdom made to the European Commission under Regulation (EC) No. 1141/2010 for renewal of an active substance

Examples

```
## Not run:

# Model definitions
sfo_sfo <- mkinmod(
  parent = mkinmod("SFO", to = "A1"),
  A1 = mkinmod("SFO"),
  use_of_ff = "max"
)

dfop_sfo <- mkinmod(
  parent = mkinmod("DFOP", to = "A1"),
  A1 = mkinmod("SFO"),
  use_of_ff = "max"
)

sfo_sfo_sfo <- mkinmod(
  parent = mkinmod("SFO", to = "A1"),
  A1 = mkinmod("SFO", to = "A2"),
  A2 = mkinmod("SFO"),
  use_of_ff = "max"
)

dfop_sfo_sfo <- mkinmod(
  parent = mkinmod("DFOP", to = "A1"),
  A1 = mkinmod("SFO", to = "A2"),
  A2 = mkinmod("SFO"),
  use_of_ff = "max"
)

d_1_2 <- lapply(experimental_data_for_UBA_2019[1:2], function(x) x$data)
names(d_1_2) <- paste("Soil", 1:2)

f_1_2_tc <- mmkin(list("DFOP-SFO-SFO" = dfop_sfo_sfo), d_1_2, error_model = "tc")

plot(f_1_2_tc, resplot = "errmod")

## End(Not run)
```

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

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

```
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

```
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

```
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 [SF0.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))
## Not run:
  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(fit.fomc$par, fit.iore$par, fit.iore.deS$par,
                  row.names = paste("model par", 1:4)))
  print(rbind(fomc = endpoints(fit.fomc)$distimes, iore = endpoints(fit.iore)$distimes,
              iore.deS = endpoints(fit.iore.deS)$distimes))

## End(Not run)
```

logistic.solution *Logistic kinetics*

Description

Function describing exponential decline from a defined starting value, with an increasing rate constant, supposedly caused by microbial growth

Usage

```
logistic.solution(t, parent.0, kmax, k0, r)
```

Arguments

t	Time.
parent.0	Starting value for the response variable at time zero.
kmax	Maximum rate constant.
k0	Minimum rate constant effective at time zero.
r	Growth rate of the increase in the rate constant.

Value

The value of the response variable at time t.

Note

The solution of the logistic model reduces to the [SF0.solution](#) if k_0 is equal to k_{max} .

References

FOCUS (2014) “Generic guidance for Estimating Persistence and Degradation Kinetics from Environmental Fate Studies on Pesticides in EU Registration” Report of the FOCUS Work Group on Degradation Kinetics, Version 1.1, 18 December 2014 <http://esdac.jrc.ec.europa.eu/projects/degradation-kinetics>

Examples

```
# Reproduce the plot on page 57 of FOCUS (2014)
plot(function(x) logistic.solution(x, 100, 0.08, 0.0001, 0.2),
      from = 0, to = 100, ylim = c(0, 100),
      xlab = "Time", ylab = "Residue")
plot(function(x) logistic.solution(x, 100, 0.08, 0.0001, 0.4),
      from = 0, to = 100, add = TRUE, lty = 2, col = 2)
plot(function(x) logistic.solution(x, 100, 0.08, 0.0001, 0.8),
      from = 0, to = 100, add = TRUE, lty = 3, col = 3)
plot(function(x) logistic.solution(x, 100, 0.08, 0.001, 0.2),
      from = 0, to = 100, add = TRUE, lty = 4, col = 4)
plot(function(x) logistic.solution(x, 100, 0.08, 0.08, 0.2),
      from = 0, to = 100, add = TRUE, lty = 5, col = 5)
legend("topright", inset = 0.05,
      legend = paste0("k0 = ", c(0.0001, 0.0001, 0.0001, 0.001, 0.08),
                     ", r = ", c(0.2, 0.4, 0.8, 0.2, 0.2)),
      lty = 1:5, col = 1:5)

# Fit with synthetic data
logistic <- mkinmod(parent = mkinmod("logistic"))

sampling_times = c(0, 1, 3, 7, 14, 28, 60, 90, 120)
parms_logistic <- c(kmax = 0.08, k0 = 0.0001, r = 0.2)
d_logistic <- mkinpredict(logistic,
  parms_logistic, c(parent = 100),
  sampling_times)
d_2_1 <- add_err(d_logistic,
  sdfunc = function(x) sigma_twocomp(x, 0.5, 0.07),
  n = 1, reps = 2, digits = 5, LOD = 0.1, seed = 123456)[[1]]

m <- mkinfit("logistic", d_2_1, quiet = TRUE)
plot_sep(m)
summary(m)$bpar
endpoints(m)$distimes
```

Description

This function simply calculates the product of the likelihood densities calculated using [dnorm](#), i.e. assuming normal distribution, with of the mean predicted by the degradation model, and the standard deviation predicted by the error model.

The total number of estimated parameters returned with the value of the likelihood is calculated as the sum of fitted degradation model parameters and the fitted error model parameters.

Usage

```
## S3 method for class 'mkinfit'  
logLik(object, ...)
```

Arguments

object	An object of class mkinfit .
...	For compatibility with the generic method

Value

An object of class [logLik](#) with the number of estimated parameters (degradation model parameters plus variance model parameters) as attribute.

Author(s)

Johannes Ranke

See Also

Compare the AIC of columns of [mmkin](#) objects using [AIC.mmkin](#).

Examples

```
## Not run:  
sfo_sfo <- mkinmod(  
  parent = mkinsub("SF0", to = "m1"),  
  m1 = mkinsub("SF0")  
)  
d_t <- FOCUS_2006_D  
f_nw <- mkinfit(sfo_sfo, d_t, quiet = TRUE) # no weighting (weights are unity)  
f_obs <- mkinfit(sfo_sfo, d_t, error_model = "obs", quiet = TRUE)  
f_tc <- mkinfit(sfo_sfo, d_t, error_model = "tc", quiet = TRUE)  
AIC(f_nw, f_obs, f_tc)  
  
## End(Not run)
```

max_twa_parent	<i>Function to calculate maximum time weighted average concentrations from kinetic models fitted with mkinfit</i>
----------------	---

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, DFOP and HS models, using the analytical formulas given in the PEC soil section of the FOCUS guidance.

Usage

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

Arguments

<code>fit</code>	An object of class <code>mkinfit</code> .
<code>windows</code>	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 `mkimod`.

`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)$bpar
endpoints(fit.1)
# k_phenol_sink is really small, therefore fix it to zero
fit.2 <- mkinfit(SFO_SFO_SFO, subset(mccall81_245T, soil == "Commerce"),
                parms.ini = c(k_phenol_sink = 0),
                fixed_parms = "k_phenol_sink", quiet = TRUE)
summary(fit.2)$bpar
endpoints(fit.1)
plot_sep(fit.2)

## End(Not run)
```

mkind	<i>A dataset class for mkin</i>
-------	---------------------------------

Description

A dataset class for mkin

Usage

```
mkind
```

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

mkinerrmin	<i>Calculate the minimum error to assume in order to pass the variance test</i>
------------	---

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 <code>mkinfit</code> .
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)
## Not run:
fit_FOCUS_E = mkinfit(SFO_SFO, FOCUS_2006_E, quiet = TRUE)
round(mkinerrmin(fit_FOCUS_E), 4)

## End(Not run)
```

mkinerrplot	<i>Function to plot squared residuals and the error model for an mkin object</i>
-------------	--

Description

This function plots the squared residuals for the specified subset of the observed variables from an `mkinfit` object. In addition, one or more dashed line(s) show the fitted error model. A combined plot of the fitted model and this error model plot can be obtained with `plot.mkinfit` using the argument `show_errrplot = TRUE`.

Usage

```
mkinerrplot(object,
  obs_vars = names(object$mkinmod$map),
  xlim = c(0, 1.1 * max(object$data$predicted)),
  xlab = "Predicted", ylab = "Squared residual",
  maxy = "auto", legend= TRUE, lpos = "topright",
  col_obs = "auto", pch_obs = "auto",
  ...)
```

Arguments

<code>object</code>	A fit represented in an <code>mkinfit</code> object.
<code>obs_vars</code>	A character vector of names of the observed variables for which residuals should be plotted. Defaults to all observed variables in the model
<code>xlim</code>	plot range in x direction.
<code>xlab</code>	Label for the x axis.
<code>ylab</code>	Label for the y axis.
<code>maxy</code>	Maximum value of the residuals. This is used for the scaling of the y axis and defaults to "auto".
<code>legend</code>	Should a legend be plotted?
<code>lpos</code>	Where should the legend be placed? Default is "topright". Will be passed on to legend .
<code>col_obs</code>	Colors for the observed variables.
<code>pch_obs</code>	Symbols to be used for the observed variables.
<code>...</code>	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

```
## Not run:
model <- mkinmod(parent = mkinsub("SF0", "m1"), m1 = mkinsub("SF0"))
fit <- mkinfit(model, FOCUS_2006_D, error_model = "tc", quiet = TRUE)
mkinerrplot(fit)

## End(Not run)
```

mkinfit

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

Description

This function maximises the likelihood of the observed data using the Port algorithm [nlminb](#), and the specified initial or fixed parameters and starting values. In each step of the optimisation, the kinetic model is solved using the function [mkinpredict](#). The parameters of the selected error model are fitted simultaneously with the degradation model parameters, as both of them are arguments of the likelihood function.

Per default, parameters in the kinetic models are internally transformed in order to better satisfy the assumption of a normal distribution of their estimators.

Usage

```
mkinfit(mkinmod, observed,
  parms.ini = "auto",
  state.ini = "auto",
  err.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",
  control = list(eval.max = 300, iter.max = 200),
  transform_rates = TRUE,
  transform_fractions = TRUE,
  quiet = FALSE,
  atol = 1e-8, rtol = 1e-10, n.outtimes = 100,
  error_model = c("const", "obs", "tc"),
  error_model_algorithm = c("d_3", "direct", "twostep", "threestep", "fourstep", "IRLS"),
  reweight.tol = 1e-8, reweight.max.iter = 10,
  trace_parms = FALSE, ...)
```

Arguments

mkinmod	A list of class <code>mkinmod</code> , containing the kinetic model to be fitted to the data, or one of the shorthand names ("SFO", "FOMC", "DFOP", "HS", "SFORB", "IORE"). If a shorthand name is given, a parent only degradation model is generated for the variable with the highest value in observed.
observed	A dataframe with the observed data. 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. Zero values in the "value" column will be removed, with a warning, in order to avoid problems with fitting the two-component error model. This is not expected to be a problem, because in general, values of zero are not observed in degradation data, because there is a lower limit of detection.
parms.ini	A named vector of initial values for the parameters, including parameters to be optimised and potentially also fixed parameters as indicated by <code>fixed_parms</code> . 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 <code>map</code> component of <code>mkinmod</code>). 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.
err.ini	A named vector of initial values for the error model parameters to be optimised. If set to "auto", initial values are set to default values. Otherwise, initial values for all error model parameters must be given.
fixed_parms	The names of parameters that should not be optimised but rather kept at the values specified in <code>parms.ini</code> .
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 <code>deSolve</code> 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 "deSolve" if a compiler is present, and "eigen" if no compiler is present and the model can be expressed using eigenvalues and eigenvectors. This argument is passed on to the helper function `mkinpredict`.

<code>method.ode</code>	The solution method passed via <code>mkinpredict</code> to <code>ode</code> in case the solution type is "deSolve". The default "lsoda" is performant, but sometimes fails to converge.
<code>use_compiled</code>	If set to FALSE, no compiled version of the <code>mkinmod</code> model is used in the calls to <code>mkinpredict</code> even if a compiled version is present.
<code>control</code>	A list of control arguments passed to <code>nlminb</code> .
<code>transform_rates</code>	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 <code>tb</code> of the HS model. If FALSE, zero is used as a lower bound for the rates in the optimisation.
<code>transform_fractions</code>	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 <code>g</code> 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 <code>ilr</code> transformation.
<code>quiet</code>	Suppress printing out the current value of the negative log-likelihood after each improvement?
<code>atol</code>	Absolute error tolerance, passed to <code>ode</code> . Default is 1e-8, lower than in <code>lsoda</code> .
<code>rtol</code>	Absolute error tolerance, passed to <code>ode</code> . Default is 1e-10, much lower than in <code>lsoda</code> .
<code>n.outtimes</code>	The length of the dataserries that is produced by the model prediction function <code>mkinpredict</code> . This impacts the accuracy of the numerical solver if that is used (see <code>solution_type</code> argument. The default value is 100.
<code>error_model</code>	If the error model is "const", a constant standard deviation is assumed. If the error model is "obs", each observed variable is assumed to have its own variance. If the error model is "tc" (two-component error model), a two component error model similar to the one described by Rocke and Lorenzato (1995) is used for setting up the likelihood function. Note that this model deviates from the model by Rocke and Lorenzato, as their model implies that the errors follow a lognormal distribution for large values, not a normal distribution as assumed by this method.
<code>error_model_algorithm</code>	If the error model is "const", the error model algorithm is ignored, because no special algorithm is needed and unweighted (also known as ordinary) least squares fitting can be applied.

The default algorithm "d_3" will directly minimize the negative log-likelihood and - independently - also use the three step algorithm described below. The fit with the higher likelihood is returned.

The algorithm "direct" will directly minimize the negative log-likelihood.

The algorithm "twostep" will minimize the negative log-likelihood after an initial unweighted least squares optimisation step.

The algorithm "threestep" starts with unweighted least squares, then optimizes only the error model using the degradation model parameters found, and then minimizes the negative log-likelihood with free degradation and error model parameters.

The algorithm "fourstep" starts with unweighted least squares, then optimizes only the error model using the degradation model parameters found, then optimizes the degradation model again with fixed error model parameters, and finally minimizes the negative log-likelihood with free degradation and error model parameters.

The algorithm "IRLS" starts with unweighted least squares, and then iterates optimization of the error model parameters and subsequent optimization of the degradation model using those error model parameters, until the error model parameters converge.

reweight.tol	Tolerance for the convergence criterion calculated from the error model parameters in IRLS fits.
reweight.max.iter	Maximum number of iterations in IRLS fits.
trace_parms	Should a trace of the parameter values be listed?
...	Further arguments that will be passed on to deSolve .

Value

A list with "mkinfit" in the class attribute. A summary can be obtained by [summary.mkinfit](#).

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

Source

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

See Also

Plotting methods [plot.mkinfit](#) and [mkinparplot](#).

Comparisons of models fitted to the same data can be made using [AIC](#) by virtue of the method [logLik.mkinfit](#).

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 mkinmod for convenience in model formulation. Pathway to sink included per default.
SFO_SFO <- mkinmod(
  parent = mkinmod("SFO", "m1"),
  m1 = mkinmod("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 = mkinmod("FOMC", "m1"),
  m1 = mkinmod("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$parms.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.obs <- mkinfit(SFO_SFO.ff, FOCUS_2006_D, error_model = "obs", quiet = TRUE)
summary(f.obs)
f.tc <- mkinfit(SFO_SFO.ff, FOCUS_2006_D, error_model = "tc", quiet = TRUE)
summary(f.tc)

## 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 <code>cfunction</code> 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 <code>use_of_ff</code> 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

```
## Not run:
model <- mkinmod(
  T245 = mkinsub("SF0", to = c("phenol"), sink = FALSE),
  phenol = mkinsub("SF0", to = c("anisolet")),
  anisolet = mkinsub("SF0"), use_of_ff = "max")
fit <- mkinfit(model, subset(mccall81_245T, soil == "Commerce"), quiet = TRUE)
mkinparplot(fit)

## End(Not run)
```

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(x, odeparms, odeini, outtimes = seq(0, 120, by = 0.1),
  solution_type = "deSolve", use_compiled = "auto", method.ode = "lsoda",
  atol = 1e-08, rtol = 1e-10, map_output = TRUE, ...)
```

Arguments

x	A kinetic model as produced by <code>mkinmod</code> , or a kinetic fit as fitted by <code>mkinfit</code> . In the latter case, the fitted parameters are used for the prediction.
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 vector 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 <code>mkinpredict</code> to <code>ode</code> 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 <code>mkinmod</code> model is used, even if is present.
atol	Absolute error tolerance, passed to <code>ode</code> . Default is 1e-8, lower than in <code>lsoda</code> .
rtol	Absolute error tolerance, passed to <code>ode</code> . Default is 1e-10, much lower than in <code>lsoda</code> .
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

```
SF0 <- mkinmod(degradinol = mkinsub("SF0"))
# Compare solution types
mkinpredict(SF0, 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,]))

## Not run:
# Predict from a fitted model
f <- mkinfit(SFO_SFO, FOCUS_2006_C)
head(mkinpredict(f))

## End(Not run)

```

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

<code>object</code>	A fit represented in an <code>mkinfit</code> object.
<code>obs_vars</code>	A character vector of names of the observed variables for which residuals should be plotted. Defaults to all observed variables in the model
<code>xlim</code>	plot range in x direction.
<code>xlab</code>	Label for the x axis. Defaults to "Time [days]".
<code>ylab</code>	Label for the y axis. Defaults to "Residual [% of applied radioactivity]".
<code>maxabs</code>	Maximum absolute value of the residuals. This is used for the scaling of the y axis and defaults to "auto".
<code>legend</code>	Should a legend be plotted? Defaults to "TRUE".
<code>lpos</code>	Where should the legend be placed? Default is "topright". Will be passed on to legend .
<code>...</code>	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("SF0", "m1"), m1 = mkinsub("SF0"))
fit <- mkinfit(model, FOCUS_2006_D, quiet = TRUE)
mkinresplot(fit, "m1")
```

mkinsub	<i>Function to set up a kinetic submodel for one state variable</i>
---------	---

Description

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

Usage

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

Arguments

<code>submodel</code>	Character vector of length one to specify the submodel type. See <code>mkimod</code> for the list of allowed submodel names.
<code>to</code>	Vector of the names of the state variable to which a transformation shall be included in the model.
<code>sink</code>	Should a pathway to sink be included in the model in addition to the pathways to other state variables?
<code>full_name</code>	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>i.e.</i> may produce unintended plot results on other operating systems or system configurations.

Value

A list for use with `mkimod`.

Author(s)

Johannes Ranke

Examples

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

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

# Now supplying full names
SFO_SFO.2 <- mkimod(
  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, i.e. with a row for each observed value, 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 `mkinfit`.

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 `mkinfite`.

Author(s)

Johannes Ranke

Examples

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

mmkin	<i>Fit one or more kinetic models with one or more state variables to one or more datasets</i>
-------	--

Description

This function calls `mkinfite` 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 <code>mkinfite</code> objects.
datasets	An optionally named list of datasets suitable as observed data for <code>mkinfite</code> .
cores	The number of cores to be used for multicore processing. This is only used when the <code>cluster</code> argument is <code>NULL</code> . On Windows machines, <code>cores > 1</code> is not supported, you need to use the <code>cluster</code> argument to use multiple logical processors.
cluster	A cluster as returned by <code>makeCluster</code> to be used for parallel execution.
...	Further arguments that will be passed to <code>mkinfite</code> .

Value

A matrix of `mkinf` 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_SF0_lin <- mkinmod(parent = mkinsub("SF0", "M1"),
  M1 = mkinsub("SF0", "M2"),
  M2 = mkinsub("SF0"), use_of_ff = "max")

m_synth_FOMC_lin <- mkinmod(parent = mkinsub("FOMC", "M1"),
  M1 = mkinsub("SF0", "M2"),
  M2 = mkinsub("SF0"), use_of_ff = "max")

models <- list(SF0_lin = m_synth_SF0_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[["SF0_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)
```

nafta	<i>Evaluate parent kinetics using the NAFTA guidance</i>
-------	--

Description

The function fits the SFO, IORE and DFOP models using `mmkin` and returns an object of class `nafta` that has methods for printing and plotting.

Usage

```
nafta(ds, title = NA, quiet = FALSE, ...)
```

Arguments

<code>ds</code>	A dataframe that must contain one variable called "time" with the time values specified by the <code>time</code> argument, one column called "name" with the grouping of the observed values, and finally one column of observed values called "value".
<code>title</code>	Optional title of the dataset
<code>quiet</code>	Should the evaluation text be shown?
<code>...</code>	Further arguments passed to <code>mmkin</code> .

Value

An list of class `nafta`. The list element named "mmkin" is the `mmkin` object containing the fits of the three models. The list element named "title" contains the title of the dataset used. The list element "data" contains the dataset used in the fits.

Author(s)

Johannes Ranke

Source

NAFTA (2011) Guidance for evaluating and calculating degradation kinetics in environmental media. NAFTA Technical Working Group on Pesticides <https://www.epa.gov/pesticide-science-and-assessing-pesticide-guidance-evaluating-and-calculating-degradation> accessed 2019-02-22

US EPA (2015) Standard Operating Procedure for Using the NAFTA Guidance to Calculate Representative Half-life Values and Characterizing Pesticide Degradation <https://www.epa.gov/pesticide-science-and-assessing-pesticide-guidance-standard-operating-procedure-using-nafta-guidance>

Examples

```
nafta_evaluation <- nafta(NAFTA_SOP_Appendix_D, cores = 1)
print(nafta_evaluation)
plot(nafta_evaluation)
```

NAFTA_SOP_2015

Example datasets from the NAFTA SOP published 2015

Description

Data taken from US EPA (2015), p. 19 and 23.

Usage

NAFTA_SOP_2015

Format

2 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

Source

NAFTA (2011) Guidance for evaluating and calculating degradation kinetics in environmental media. NAFTA Technical Working Group on Pesticides <https://www.epa.gov/pesticide-science-and-assessing-pesticide-guidance-evaluating-and-calculating-degradation> accessed 2019-02-22

US EPA (2015) Standard Operating Procedure for Using the NAFTA Guidance to Calculate Representative Half-life Values and Characterizing Pesticide Degradation <https://www.epa.gov/pesticide-science-and-assessing-pesticide-science-standard-operating-procedure-using-nafta-guidance>

Examples

```
nafta_evaluation <- nafta(NAFTA_SOP_Appendix_D, cores = 1)
print(nafta_evaluation)
plot(nafta_evaluation)
```

NAFTA_SOP_Attachment

Example datasets from Attachment 1 to the NAFTA SOP published 2015

Description

Data taken from from Attachment 1 of the SOP.

Usage

NAFTA_SOP_Attachment

Format

A list (NAFTA_SOP_Attachment) containing 16 datasets suitable for the evaluation with `nafta`

Source

NAFTA (2011) Guidance for evaluating and calculating degradation kinetics in environmental media. NAFTA Technical Working Group on Pesticides <https://www.epa.gov/pesticide-science-and-assessing-pesticide-guidance-evaluating-and-calculating-degradation> accessed 2019-02-22

US EPA (2015) Standard Operating Procedure for Using the NAFTA Guidance to Calculate Representative Half-life Values and Characterizing Pesticide Degradation <https://www.epa.gov/pesticide-science-and-assessing-pesticide-guidance-standard-operating-procedure-using-nafta-guidance>

Examples

```
nafta_att_p5a <- nafta(NAFTA_SOP_Attachment[["p5a"]], cores = 1)
print(nafta_att_p5a)
plot(nafta_att_p5a)
```

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,
     show_errplot = 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 <code>mkinfit</code> .
obs_vars	A character vector of names of the observed variables for which the data and the model should be plotted. Defaults 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.
show_errplot	Should squared residuals and the error model be shown? If only one plot of the fits is shown, this 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 <code>legend</code> as the first argument. If not length one, this should be of the same length as the <code>obs_var</code> argument.
inset	Passed to <code>legend</code> 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 <code>plot</code> .

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
## Not run:
SFO_SFO <- mkinmod(parent = mkinsub("SFO", "m1", full = "Parent"),
                  m1 = mkinsub("SFO", full = "Metabolite M1" ))
fit <- mkinfit(SFO_SFO, FOCUS_2006_D, quiet = TRUE, error_model = "tc")
plot(fit)
plot(fit, show_residuals = TRUE)
plot(fit, show_errplot = 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"))

# Show the observed variables separately, with the error model
plot(fit, sep_obs = TRUE, show_errplot = TRUE, lpos = c("topright", "bottomright"),
     show_errmin = TRUE)

## End(Not run)

```

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 χ^2 error level.

Usage

```

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

```

Arguments

x	An object of class <code>mmkin</code> , 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.
resplot	Should the residuals plotted against time, using <code>mkkinresplot</code> , or as squared residuals against predicted values, with the error model, using <code>mkkinerrplot</code> .
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 <code>mtext</code> .
rel.height.middle	The relative height of the middle plot, if more than two rows of plots are shown.
...	Further arguments passed to <code>plot.mkinfit</code> and <code>mkkinresplot</code> .

Value

The function is called for its side effect.

Author(s)

Johannes Ranke

Examples

```
## Not run:
# Only use one core not to offend CRAN checks
fits <- mmkin(c("FOMC", "HS"),
             list("FOCUS B" = FOCUS_2006_B, "FOCUS C" = FOCUS_2006_C), # named list for titles
             cores = 1, quiet = TRUE, error_model = "tc")
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])

# Show the error models
plot(fits["FOMC", ], resplot = "errmod")

## End(Not run)
```

plot.nafta	<i>Plot the results of the three models used in the NAFTA scheme. The plots are ordered with increasing complexity of the model in this function (SFO, then IORE, then DFOP).</i>
------------	---

Description

Calls `plot.mmkin`.

Usage

```
## S3 method for class 'nafta'
plot(x, legend = FALSE, main = "auto", ...)
```

Arguments

x	An object of class <code>nafta</code> .
legend	Should a legend be added?
main	Possibility to override the main title of the plot.
...	Further arguments passed to <code>plot.mmkin</code> .

Value

The function is called for its side effect.

Author(s)

Johannes Ranke

print.mkinds	<i>Print mkinds objects</i>
--------------	-----------------------------

Description

Print mkinds objects.

Usage

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

Arguments

x	An <code>mkinds</code> 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_SF0_lin <- mkinmod(parent = list(type = "SF0", to = "M1"),
                           M1 = list(type = "SF0", to = "M2"),
                           M2 = list(type = "SF0", use_of_ff = "max"))

print(m_synth_SF0_lin)
```

```
print.nafta          Print nafta objects
```

Description

Print nafta objects. The results for the three models are printed in the order of increasing model complexity, i.e. SFO, then IORE, and finally DFOP.

Usage

```
## S3 method for class 'nafta'
print(x, quiet = TRUE, digits = 3, ...)
```

Arguments

```
x                An nafta object.
quiet            Should the evaluation text be shown?
digits          Number of digits to be used for printing parameters and dissipation times.
...             Not used.
```

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

```
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	<i>Single First-Order kinetics</i>
--------------	------------------------------------

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

<code>t</code>	Time.
<code>parent.0</code>	Starting value for the response variable at time zero.
<code>k_12</code>	Kinetic constant describing transfer from free to bound.
<code>k_21</code>	Kinetic constant describing transfer from bound to free.
<code>k_1output</code>	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)
```

sigma_twocomp	<i>Two component error model</i>
---------------	----------------------------------

Description

Function describing the standard deviation of the measurement error in dependence of the measured value y :

$$\sigma = \sqrt{\sigma_{low}^2 + y^2 * rsd_{high}^2}$$

This is the error model used for example by Werner et al. (1978). The model proposed by Rocke and Lorenzato (1995) can be written in this form as well, but assumes approximate lognormal distribution of errors for high values of y .

Usage

```
sigma_twocomp(y, sigma_low, rsd_high)
```

Arguments

y	The magnitude of the observed value
sigma_low	The asymptotic minimum of the standard deviation for low observed values
rsd_high	The coefficient describing the increase of the standard deviation with the magnitude of the observed value

Value

The standard deviation of the response variable.

References

Werner, Mario, Brooks, Samuel H., and Knott, Lancaster B. (1978) Additive, Multiplicative, and Mixed Analytical Errors. *Clinical Chemistry* 24(11), 1895-1898.

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

summary.mkinfit	<i>Summary method for class "mkinfit"</i>
-----------------	---

Description

Lists model equations, initial parameter values, optimised parameters with some uncertainty statistics, the chi2 error levels calculated according to FOCUS guidance (2006) as defined therein, formation fractions, DT50 values 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 <code>mkinfit</code> .
x	an object of class <code>summary.mkinfit</code> .
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 <code>print</code> .

Value

The summary function returns a list 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 <code>summary.modFit</code>
cov.unscaled, cov.scaled, info, niter, stopmess, par	As in <code>summary.modFit</code>
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.
errparms	Error model parameters.
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. Note that this is a simplified version of the error model proposed by Rocke and Lorenzato (1995), as in their model the error of the measured values approximates lognormal distribution for high values, whereas we are using normally distributed error components all along.

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 twelve datasets as an R6 class defined by `mkind`s, each containing, 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 `mkinf`

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_SF0_par <- mkinpredict(m_synth_SF0_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("SF0_lin", "SF0_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 simplified version of 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_SF0_lin, synthetic_data_for_UBA_2014[[1]]$data,
  quiet = TRUE)
plot_sep(fit)
summary(fit)

## End(Not run)

```

test_data_from_UBA_2014

Three experimental datasets from two water sediment systems and one soil

Description

The datasets were used for the comparative validation of several kinetic evaluation software packages (Ranke, 2014).

Usage

```
test_data_from_UBA_2014
```

Format

A list containing three datasets as an R6 class defined by `mkind`s. Each dataset has, among others, the following components

`title` The name of the dataset, e.g. UBA_2014_WS_river

`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

Examples

```
## Not run:
# This is a level P-II evaluation of the dataset according to the FOCUS kinetics
# guidance. Due to the strong correlation of the parameter estimates, the
# covariance matrix is not returned. Note that level P-II evaluations are
# generally considered deprecated due to the frequent occurrence of such
# large parameter correlations, among other reasons (e.g. the adequacy of the
# model).
m_ws <- mkinmod(parent_w = mkinsub("SFO", "parent_s"),
               parent_s = mkinsub("SFO", "parent_w"))
f_river <- mkinfit(m_ws, test_data_from_UBA_2014[[1]]$data, quiet = TRUE)
plot_sep(f_river)

summary(f_river)$bpar
mkinerrmin(f_river)

# This is the evaluation used for the validation of software packages
# in the expertise from 2014
m_soil <- mkinmod(parent = mkinsub("SFO", c("M1", "M2")),
                 M1 = mkinsub("SFO", "M3"),
                 M2 = mkinsub("SFO", "M3"),
                 M3 = mkinsub("SFO"),
                 use_of_ff = "max")

f_soil <- mkinfit(m_soil, test_data_from_UBA_2014[[3]]$data, quiet = TRUE)
plot_sep(f_soil, lpos = c("topright", "topright", "topright", "bottomright"))
summary(f_soil)$bpar
mkinerrmin(f_soil)

## End(Not run)
```

transform_odeparms	<i>Functions to transform and backtransform kinetic parameters for fitting</i>
--------------------	--

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 parameters that can only take on positive values, a simple log transformation is used. For compositional parameters, such as the formation 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 <code>mkinmod</code> , containing the names of the model variables that are needed for grouping the formation fractions before <code>ilr</code> 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 <code>ilr</code> 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)
fit.s <- summary(fit)
# Transformed and backtransformed parameters
print(fit.s$par, 3)
print(fit.s$bpar, 3)
```

```

## Not run:
# Compare to the version without transforming rate parameters
fit.2 <- mkinfit(SFO_SFO, FOCUS_2006_D, transform_rates = FALSE, quiet = TRUE)
fit.2.s <- summary(fit.2)
print(fit.2.s$par, 3)
print(fit.2.s$bpar, 3)

## 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)
fit.ff.s <- summary(fit.ff)
print(fit.ff.s$par, 3)
print(fit.ff.s$bpar, 3)
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)
fit.ff.2.s <- summary(fit.ff.2)
print(fit.ff.2.s$par, 3)
print(fit.ff.2.s$bpar, 3)

## 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 <code>mmkin</code> 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 <code>mmkin</code> object, otherwise either a list of <code>mkinfit</code> objects or a single <code>mkinfit</code> 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"]]  
)
```

Index

*Topic **datasets**

- experimental_data_for_UBA_2019, 8
- FOCUS_2006_datasets, 11
- FOCUS_2006_DFOP_ref_A_to_B, 11
- FOCUS_2006_FOMC_ref_A_to_F, 12
- FOCUS_2006_HS_ref_A_to_F, 13
- FOCUS_2006_SFO_ref_A_to_F, 14
- mccall181_245T, 23
- mkin, 24
- NAFTA_SOP_2015, 44
- NAFTA_SOP_Attachment, 44
- schaefer07_complex_case, 51
- synthetic_data_for_UBA_2014, 56
- test_data_from_UBA_2014, 59

*Topic **hplot**

- mkinerrplot, 26
- mkinparplot, 34
- mkinresplot, 37

*Topic **manip**

- add_err, 3
- DFOP.solution, 7
- endpoints, 8
- FOMC.solution, 15
- geometric_mean, 16
- HS.solution, 16
- ilr, 17
- IORE.solution, 18
- logistic.solution, 19
- max_twa_parent, 22
- mkin_long_to_wide, 40
- mkin_wide_to_long, 40
- mkinerrmin, 24
- mkinpredict, 35
- SFO.solution, 52
- SFORB.solution, 53
- transform_odeparms, 60

*Topic **models**

- mkinmod, 32

*Topic **optimize**

- mkinfit, 27

- mmkin, 41

*Topic **utilities**

- summary.mkinfit, 55

- [.mmkin, 42, 47, 62

- add_err, 3

- AIC, 31

- AIC.mmkin, 5, 21

- backtransform_odeparms
(transform_odeparms), 60

- CAKE_export, 6

- cfunction, 33

- deSolve, 28, 30

- DFOP.solution, 7

- dnorm, 21

- endpoints, 8

- experimental_data_for_UBA_2019, 8

- FOCUS_2006_A (FOCUS_2006_datasets), 11

- FOCUS_2006_B (FOCUS_2006_datasets), 11

- FOCUS_2006_C (FOCUS_2006_datasets), 11

- FOCUS_2006_D (FOCUS_2006_datasets), 11

- FOCUS_2006_datasets, 11

- FOCUS_2006_DFOP_ref_A_to_B, 11

- FOCUS_2006_E (FOCUS_2006_datasets), 11

- FOCUS_2006_F (FOCUS_2006_datasets), 11

- FOCUS_2006_FOMC_ref_A_to_F, 12

- FOCUS_2006_HS_ref_A_to_F, 13

- FOCUS_2006_SFO_ref_A_to_F, 14

- FOMC.solution, 15

- geometric_mean, 16

- HS.solution, 16

- ilr, 17, 29, 60, 61

invilr(ilr), 17
IORE.solution, 18

legend, 26, 38, 46
logistic.solution, 19
logLik, 21
logLik.mkinfit, 20, 31
lsoda, 29, 36

makeCluster, 41
max_twa_parent, 22
mccall81_245T, 23
mkin_long_to_wide, 40
mkin_wide_to_long, 40
mkininds, 9, 24, 49, 57, 59
mkinerrmin, 24
mkinerrplot, 26, 48
mkinfit, 3, 6, 8, 9, 21, 22, 25, 26, 27, 33–36, 38, 40–42, 45, 46, 55, 57, 59, 60
mkinmod, 23, 28, 29, 32, 35, 36, 39, 41, 50, 61
mkinparplot, 31, 34
mkinplot, 27, 35, 38
mkinpredict, 3, 27, 29, 35, 36
mkinresplot, 37, 48
mkinsub, 39
mmkin, 3, 5, 21, 31, 41, 43, 48, 63
mtext, 48

nafta, 43, 45, 49, 50
NAFTA_SOP_2015, 44
NAFTA_SOP_Appendix_B (NAFTA_SOP_2015), 44
NAFTA_SOP_Appendix_D (NAFTA_SOP_2015), 44
NAFTA_SOP_Attachment, 44
nlminb, 27, 29

ode, 29, 36

plot, 26, 38, 46
plot.mkinfit, 26, 31, 35, 38, 45, 48
plot.mmkin, 42, 47, 49
plot.nafta, 49
plot_sep(plot.mkinfit), 45
print.mkininds, 49
print.mkinmod, 50
print.nafta, 50
print.summary.mkinfit
 (summary.mkinfit), 55

R6Class, 24

schaefer07_complex_case, 51
schaefer07_complex_results
 (schaefer07_complex_case), 51
SFO.solution, 15, 19, 20, 52
SFORB.solution, 53
sigma_twocomp, 54
summary.mkinfit, 8, 25, 30, 55
synthetic_data_for_UBA_2014, 56

test_data_from_UBA_2014, 59
tikz, 45, 47
transform_odeparms, 60