Package 'ArArRedux'

January 20, 2025

Title Rigorous Data Reduction and Error Propagation of Ar40 / Ar39 Data

Version 1.0

Date 2018-08-13

Description Processes noble gas mass spectrometer data to determine the isotopic composition of argon (comprised of Ar36, Ar37, Ar38, Ar39 and Ar40) released from neutron-irradiated potassium-bearing minerals. Then uses these compositions to calculate precise and accurate geochronological ages for multiple samples as well as the covariances between them. Error propagation is done in matrix form, which jointly treats all samples and all isotopes simultaneously at every step of the data reduction process. Includes methods for regression of the time-resolved mass spectrometer signals to t=0 ('time zero') for both single- and multi-collector instruments, blank correction, mass fractionation correction, detector intercalibration, decay corrections, interference corrections, interpolation of the irradiation parameter between neutron fluence monitors, and (weighted mean) age calculation. All operations are performed on the logs of the ratios between the different argon isotopes so as to properly treat them as 'compositional data', sensu Aitchison [1986, The Statistics of Compositional Data, Chapman and Hall].

Author Pieter Vermeesch [aut, cre]

Maintainer Pieter Vermeesch <p.vermeesch@ucl.ac.uk>

Depends R (>= 3.0.2)

Imports utils, stats, methods, graphics, grDevices

License GPL-2

LazyData true

NeedsCompilation no

Repository CRAN

RoxygenNote 6.0.1

Date/Publication 2018-10-03 21:52:31 UTC

Contents

`average` *Calculate the arithmetic mean*

Description

Calculate the arithmetic mean of some logratio data

Usage

```
average(x, i = NULL, newlabel = NULL)
```

Arguments

x	an object of class redux or logratios
i	(optional) vector of sample indices
newlabel	(optional) string with the new label to be assigned to the averaged values

Value

an object of the same class as x

Examples

```
data(Melbourne)
K <- average(Melbourne$X,grep("K:",Melbourne$X$labels),newlabel="K-glass")
plotcorr(K)
```

---

averagebyday *Average all the data collected on the same day.*

Description

This function is useful for grouping a number of replicate air shots or calibration experiments

Usage

```
averagebyday(x, newlabel)
```

Arguments

x	an object of class timeresolved, logratios, PHdata or redux
newlabel	a string with the new label that should be given to the average

Value

an object of the same class as x

Examples

```
dfile <- system.file("Calibration.csv",package="ArArRedux")
dlabels <- c("H1","AX","L1","L2")
md <- loaddata(dfile,dlabels,PH=TRUE)
ld <- fitlogratios(blankcorr(md))
d <- averagebyday(ld,"DCAL")
plotcorr(d)
```

blankcorr *Apply a blank correction*

Description

Applies a blank correction to some time-resolved mass spectrometer data

Usage

```
blankcorr(x, ...)

## Default S3 method:
blankcorr(x, ...)

## S3 method for class 'timeresolved'
blankcorr(x, blanklabel = NULL, prefix = "", ...)

## S3 method for class 'PHdata'
blankcorr(x, blanklabel = NULL, prefix = "", ...)
```

Arguments

x	an object of class timeresolved or PHdata
...	other arguments
blanklabel	as string denoting the prefix of the blanks
prefix	a string to be prepended to the non-blanks

Value

an object of class blankcorrected

Examples

```
samplefile <- system.file("Samples.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
m <- loaddata(samplefile,masses) # samples and J-standards
blanklabel <- "EXB#"
l <- fitlogratios(blankcorr(m,blanklabel),"Ar40")
plotcorr(l)
```

blankcorrected *The* blankcorrected *class*

Description

An object class containing blank-corrected mass spectrometry data

Details

Extends the class classes timeresolved and PHdata by adding an additional list item blankindices containg the index of the nearest blank. fitlogratios uses this information to group the samples during regression to 'time zero'.

calibration *Detector calibration*

Description

Apply the detector calibration for multicollector data

Usage

```
calibration(X, clabel)
```

Arguments

X	an object of class redux
clabel	the label of the detector calibration data

Value

an object of class redux

Examples

```
data(Melbourne)
C <- calibration(Melbourne$X,"DCAL")
plotcorr(C)
```

clcorrection *Cl-interference correction*

Description

Apply the interference correction for the Cl-decay products

Usage

```
clcorrection(X, irr)
```

Arguments

X	an object of class redux
irr	the irradiation schedule

Value

an object of class redux

Examples

```
data(Melbourne)
Cl <- clcorrection(Melbourne$X,Melbourne$irr)
plotcorr(Cl)
```

concat *Merge a list of logratio data*

Description

Recursively concatenates a list of logratio data into one big dataset

Usage

```
concat(lrlist)
```

Arguments

lrlist	a list containing items of class logratios or redux

Value

an object of the same class as x containing the merged dataset

Examples

```
samplefile <-  system.file("Samples.csv",package="ArArRedux")
kfile <- system.file("K-glass.csv",package="ArArRedux")
cafile <- system.file("Ca-salt.csv",package="ArArRedux")
dfile <- system.file("Calibration.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
blanklabel <- "EXB#"
Jpos <- c(3,15)
dlabels <- c("H1","AX","L1","L2")

m <- loaddata(samplefile,masses) # samples and J-standards
mk <- loaddata(kfile,masses) # K-interference data
mca <- loaddata(cafile,masses) # Ca interference data
md <- loaddata(dfile,dlabels,PH=TRUE) # detector intercalibrations

# form and fit logratios
l <- fitlogratios(blankcorr(m,blanklabel),"Ar40")
lk <- fitlogratios(blankcorr(mk,blanklabel,"K:"),"Ar40")
k <- getmasses(lk,"Ar39","Ar40") # subset on the relevant isotopes
lca <- fitlogratios(blankcorr(mca,blanklabel,"Ca:"),"Ar37")
ca <- getmasses(lca,c("Ar36","Ar39"),c("Ar37","Ar37")) # subset
ld <- fitlogratios(blankcorr(md))
d <- averagebyday(ld,"DCAL")

# merge all data (except air shots) into one big logratio structure
X <- newredux(concat(list(l,k,ca,d)),Jpos)
data(Melbourne)
if (isTRUE(all.equal(Melbourne$X,X))) {
   print("We just reconstructed the built-in dataset Melbourne$X")}
```

`decaycorrection` *Correct for radioactive decay occurred since irradiation*

Description

Correct for radioactive decay of neutron-induced 37Ar and 39Ar occurred since irradiation

Usage

```
decaycorrection(X, irr, isotope)
```

Arguments

`X`	an objects of class `redux`
`irr`	the irradiation schedule
`isotope`	a string denoting the isotope that needs correcting

Value

an object of class redux

Examples

```
data(Melbourne)
C <- calibration(Melbourne$X,"DCAL")
A <- massfractionation(C,Melbourne$fract)
D9 <- decaycorrection(A,Melbourne$irr,"Ar39")
plotcorr(D9)
```

---

fitlogratios *Extrapolation to 'time zero'*

Description

This function extrapolates time resolved mass spectrometer data to t=0. When fed with multicollector data, it forms the ratios of the raw signals, forms their logs and performs linear regression to t=0 When fed with single collector data, the function first takes their logs and extrapolates them to t=0 before taking ratios, unless denmass=NULL, in which case the logs of the raw signals are extrapolated.

Usage

```
fitlogratios(x, ...)

## Default S3 method:
fitlogratios(x, ...)

## S3 method for class 'timeresolved'
fitlogratios(x, denmass, ...)

## S3 method for class 'PHdata'
fitlogratios(x, denmass = NULL, ...)
```

Arguments

x	an object of class timeresolved or PHdata
...	further arguments (see below)
denmass	a string denoting the denominator isotope

Value

an object of class logratios

Examples

```
samplefile <- system.file("Samples.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
m <- loaddata(samplefile,masses) # samples and J-standards
blanklabel <- "EXB#"
l <- fitlogratios(blankcorr(m,blanklabel),"Ar40")
plotcorr(l)
```

`fractionation` *Compute the mass fractionation correction*

Description

Compares the measured 40Ar/36Ar ratio of an air shot on a given detector with the atmospheric ratio.

Usage

```
fractionation(fname, detector, MS = "ARGUS-VI", PH = FALSE)
```

Arguments

`fname`	a .csv file with the air shot data
`detector`	the name of the ion detector
`MS`	the type of mass spectrometer
`PH`	TRUE if the data were recorded in 'peak hopping' mode, FALSE if recorded in multicollector mode.

Value

an object of class `logratios`

Examples

```
data(Melbourne)
fd37file <- system.file("AirL2.csv",package="ArArRedux")
fd40file <- system.file("AirH1.csv",package="ArArRedux")
fract <- list(fractionation(fd37file,"L2",PH=TRUE),
              fractionation(fd40file,"H1",PH=FALSE))
if (isTRUE(all.equal(Melbourne$fract,fract))){
  print("We just re-created the fractionation correction for the Melbourne dataset")
}
```

get4039 *Calculate the 40Ar*/39ArK-ratios*

Description

Calculate the 40Ar*/39ArK-ratios of interference corrected logratio intercept data

Usage

```
get4039(X, irr)
```

Arguments

X	an object of class redux containing some interference corrected logratio intercept data
irr	the irradiation schedule

Value

an object of class link{redux} containing the 40Ar*/39ArK-ratios as intercepts and its covariance matrix as covmat

Examples

```
data(Melbourne)
R <- get4039(Melbourne$X,Melbourne$irr)
plotcorr(R)
```

getages *Calculate 40Ar/39Ar ages*

Description

Calculate 40Ar/39Ar ages from a vector of 40Ar/39Ar-ratios and J-factors

Usage

```
getages(RJ)
```

Arguments

RJ	an object of class Redux containing the amalgamated ^{40}Ar*/^{39}Ar$_K$-ratios and J-factors with their covariance matrix

Value

an object of class results containing the ages and their covariance matrix

Examples

```
data(Melbourne)
R <- get4039(Melbourne$X,Melbourne$irr)
J <- getJfactors(R)
ages <- getages(J)
plotcorr(ages)
```

`getJfactors` *Calculate the irradiation parameter ('J factor')*

Description

Interpolate the irradiation parameters for the samples given the 40Ar*/39ArK ratios of the samples and fluence monitors

Usage

```
getJfactors(R)
```

Arguments

`R`	a vector of 40Ar*/39ArK ratios

Value

an object of class `redux` containing, as `intercepts`, the 40Ar*/39ArK ratios of the samples, the interpolated J-factors, and the 40K decay constant; and as `covmat`: the covariance matrix. All other class properties are inherited from `R`.

Examples

```
data(Melbourne)
R <- get4039(Melbourne$X,Melbourne$irr)
J <- getJfactors(R)
plotcorr(J)
```

`getmasses` *Select a subset of isotopes from a dataset*

Description

Extracts the intercepts, covariance matrix, etc. of a selection of isotopes from a larger dataset

Usage

```
getmasses(x, ...)

## Default S3 method:
getmasses(x, ...)

## S3 method for class 'timeresolved'
getmasses(x, mass, invert = FALSE, ...)

## S3 method for class 'logratios'
getmasses(x, num, den, invert = FALSE, ...)

## S3 method for class 'redux'
getmasses(x, num, den, invert = FALSE, ...)
```

Arguments

x	an object of class logratios, timeresolved, PHdata or redux.
...	other arguments
mass	a vector of strings denoting the masses of interest
invert	boolean parameter indicating whether the selection should be inverted (default = FALSE)
num	vector of strings indicating the numerator isotopes
den	vector of string indicating the denominator isotopes

Value

an object of the same class as x

Examples

```
kfile <- system.file("K-glass.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
mk <- loaddata(kfile,masses)
lk <- fitlogratios(blankcorr(mk,"EXB#","K:"),"Ar40")
k <- getmasses(lk,"Ar39","Ar40") # subset of the relevant isotopes
plotcorr(k)
```

---

interference *define the interference corrections*

Description

create a new object of class logratios containing the interferences from neutron reactions on Ca and K

Usage

```
interference(intercepts, covmat, num, den, irr, label)
```

Arguments

intercepts	a vector with logratios
covmat	the covariance matrix of the logratios
num	a vector of strings marking the numerator isotopes of intercepts
den	a vector of strings marking the denominator isotopes of intercepts
irr	an object of class irradiations
label	a string with a name which can be used to identify the interference data in subsequent calculations

Value

an object of class logratios

Examples

```
samplefile <- system.file("Samples.csv",package="ArArRedux")
irrfile <- system.file("irradiations.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
X <- read(samplefile,masses,blabel="EXB#",Jpos=c(3,15))
irr <- loadirradiations(irrfile)
# assume log(36Ar/37Ar) = log(39Ar/37Ar) = 1 in co-irradiate Ca-salt
# with variances of 0.0001 and zero covariances
ca <- interference(intercepts=c(1,1),
                   covmat=matrix(c(0.001,0,0,0.001),nrow=2),
                   num=c("Ar39","Ar36"),den=c("Ar37","Ar37"),
                   irr=X$irr[1],label="Ca-salt")
# assume log(39Ar/40Ar) = 4.637788 in co-irradiate K-glass
# with variance 7.9817e-4
k <- interference(intercepts=4.637788,covmat=7.9817e-4,
                  num="Ar39",den="Ar40",irr=X$irr[1],
                  label="K-glass")
ages <- process(X,irr,ca=ca,k=k)
summary(ages)
```

---

loaddata *Load mass spectrometer data*

Description

Loads a .csv file with raw mass spectrometer data

Usage

```
loaddata(fname, masses, MS = "ARGUS-VI", PH = FALSE)
```

Arguments

fname	the file name, must end with .csv
masses	a vector of strings denoting the order of the isotopes listed in the table
MS	the type of mass spectrometer
PH	a boolean indicating whether the data are to be treated as multicollector (PH=FALSE) or 'peak hopping' (PH=TRUE) data. The default is PH=FALSE.

Value

if PH=FALSE: an object of class timeresolved
if PH=TRUE: an object of class PHdata

Examples

```
samplefile <- system.file("Samples.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
m <- loaddata(samplefile,masses) # samples and J-standards
plot(m,"MD2-1a","Ar40")
```

---

loadirradiations *Load the irradiation schedule*

Description

Loads a .csv file with the schedule of a multi-stage neutron irradiation

Usage

```
loadirradiations(fname)
```

Arguments

fname	file name (in .csv format)

Value

a list of irradiations, where each irradiation is a named list containing:

tin: vector with the start times of irradiations
tout: vector with the end times of irradiations
P: vector with the power of the irradiations

Examples

```
irrfile <- system.file("irradiations.csv",package="ArArRedux")
irr <- loadirradiations(irrfile)
str(irr)
```

logratios *The* logratios *class*

Description

An object class containing logratio intercepts

Details

A list with the following items:

`labels`: a vector of strings denoting the names of the runs
`num`: a vector of strings denoting the numerator isotopes
`den`: a vector of strings denoting the denominator isotopes
`intercepts`: a vector of logratio intercepts or values
`covmat`: the covariance matrix of `intercepts`
`irr`: a vector of strings denoting the irradiation runs
`pos`: a vector of integers with the positions in the irradiation stack
`thedate`: a vector containing the acquisition dates and times
`nlr`: a vector with the number of logratios per run

massfractionation *Apply the mass fractionation correction*

Description

Applies the fractionation obtained from air shot data by `fractionation` to the denominator detector in order to correct it for the mass difference between the numerator and denominator isotopes.

Usage

```
massfractionation(X, fract)
```

Arguments

`X`	an object of class `redux`
`fract`	a list with fractionation data for Ar37, Ar39 and Ar40

Value

an object of class `redux`

Examples

```
data(Melbourne)
C <- calibration(Melbourne$X,"DCAL")
A <- massfractionation(C,Melbourne$fract)
plotcorr(A)
```

`Melbourne` *An example dataset*

Description

Contains all the relevant information needed for the data reduction some ARGUS-IV data from the University of Melbourne

Author(s)

David Philips <dphillip@unimelb.edu.au>

Examples

```
data(Melbourne)
plotcorr(Melbourne$X)
```

`newredux` *Create a new* `redux` *object*

Description

Initialises a new `redux` object by packing a `logratios` dataset together with all the parameters needed for age calculation

Usage

```
newredux(X, Jpos, detectors = list(Ar36 = "H1", Ar37 = "L2", Ar38 = "L1", Ar39
  = "AX", Ar40 = "H1"))
```

Arguments

`X`	an object of class `logratios`
`Jpos`	a vector of integers denoting the positions of the fluence monitors in the irradiation stack
`detectors`	a list of strings denoting the detectors for each argon isotope

Value

an object of class `redux`

param *Set or get Ar-Ar_Redux parameters*

Description

This function is used to query and modify the half lives, standard ages etc. associated with an object of class redux

Usage

```
param(X, ...)
```

Arguments

X	an object of class redux
...	any combination of the parameters given below

Details

param grants access to the following parameters:

l0: 40K decay constant (default value = 5.5492e-4 Ma-1, Renne et al. [2010])
sl0: standard error of the 40K decay constant (default value = 0.0047e-4 Ma-1)
l7: 37Ar decay constant (default value = 7.2438 yr-1, Renne and Norman [2001])
sl7: standard error of the 37Ar decay constant (default value = 0.0083 yr-1)
l9: 39Ar decay constant (0.002577 yr-1 Stoenner et al. [1965])
sl9: standard error of the 39Ar decay constant (0.000014 yr-1)
l6: 36Cl decay constant (default value = 2301.3e-9 yr-1)
sl6: standard error of the 36Cl decay constant (default value = 7.6e-9 yr-1
pcl: (36Cl/38Cl)-production rate (default value = 252.7 for OSTR reactor, Renne et al. [2008])
spcl: standard error of the (36Cl/38Cl)-production rate (default value = 1.8)
ts: age of the fluence monitor (default = 28.201 Myr for the Fish Canyon Tuff, Kuiper et al. [2008])
sts: standard error of the fluence monitor age (default value = 0.023 Myr)
air: atmospheric 40Ar/36Ar ratio (default value = 298.56, Lee et al. [2006])
sair: standard error of the atmospheric 40Ar/36Ar ratio (default value = 0.155)

Value

returns the modified redux object OR the current parameter values if no optional arguments are supplied.

Examples

```
data(Melbourne)
param(Melbourne$X)$air
Y <- param(Melbourne$X,air=295.5)
param(Y)$air
```

PHdata *The* PHdata *class*

Description

An object class containing time resolved 'peak-hopping' mass spectrometry data

Details

A list with the following items:

`masses`: a vector of strings denoting the isotopes monitored in each run
`signals`: a list with objects of class `timeresolved`, each corresponding to a detector (i.e. length(`signals`)==1 for single collector instruments).

See Also

`loaddata`

plot.timeresolved *Plot a time resolved mass spectrometry signal*

Description

Plots the raw signal of a given isotope against time.

Usage

```
## S3 method for class 'timeresolved'
plot(x, label, mass, ...)

## S3 method for class 'PHdata'
plot(x, label, mass, ...)
```

Arguments

`x`	an object of class `timeresolved` or `PHdata`
`label`	a string with the name of the run
`mass`	a string indicating the isotope of interest
`...`	optional parameters

Examples

```
samplefile <- system.file("Samples.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
mMC <- loaddata(samplefile,masses)
plot(mMC,"MD2-1a","Ar40")
mPH <- loaddata(samplefile,masses,PH=TRUE)
plot(mPH,"MD2-1a","Ar40")
```

---

plotcorr *Plot a matrix with correlation coefficients*

Description

Converts the covariance matrix to a correlation matrix and plots this is a coloured image for visual inspection.

Usage

```
plotcorr(X)
```

Arguments

X	a data structure (list) containing an item called 'covmat' (covariance matrix)

Examples

```
data(Melbourne)
plotcorr(Melbourne$X)
```

---

process *Process logratio data and calculate 40Ar/39Ar ages*

Description

Performs detector calibration, mass fractionation correction, decay corrections, interference corrections, interpolates J-factors and calculates ages.

Usage

```
process(X, irr, fract = NULL, ca = NULL, k = NULL)
```

Arguments

X	an object of class redux
irr	the irradiation schedule
fract	list with air shot data (one item per denominator isotope)
ca	an object of class logratios with Ca-interference data (not necessary if interferences are included in X)
k	an object of class logratios with K-interference data (not necessary if interferences are included in X)

Examples

```
data(Melbourne)
ages <- process(Melbourne$X,Melbourne$irr,Melbourne$fract)
summary(ages)
```

---

read *Read mass spectrometer data*

Description

Reads raw mass spectrometer data and parses it into a redux format for further processing.

Usage

```
read(xfile, masses, blabel, Jpos, kfile = NULL, cafile = NULL,
  dfile = NULL, dlabels = NULL, MS = "ARGUS-VI")
```

Arguments

xfile	a .csv file with samples and fluence monitor data
masses	a list which specifies the order in which the isotopes are recorded by the mass spectrometer
blabel	a prefix string denoting the blanks
Jpos	a vector of integers denoting the positions of the fluence monitors in the irradiation stack
kfile	a .csv file with the K-interference monitor data (optional)
cafile	a .csv file with the Ca-interference monitor data (optional)
dfile	a .csv file with the detector calibration data (optional)
dlabels	a list which specifies the names of the detectors and the order in which they were recorded by the mass spectrometer
MS	a string denoting the type of mass spectrometer. At the moment only the ARGUS-IV is listed. Please contact the author to add other file formats to Ar-Ar_Redux.

Value

an object of class redux.

Examples

```
samplefile <-  system.file("Samples.csv",package="ArArRedux")
kfile <- system.file("K-glass.csv",package="ArArRedux")
cafile <- system.file("Ca-salt.csv",package="ArArRedux")
dfile <- system.file("Calibration.csv",package="ArArRedux")
masses <- c("Ar37","Ar38","Ar39","Ar40","Ar36")
dlabels <- c("H1","AX","L1","L2")
X <- read(samplefile,masses,blabel="EXB#",Jpos=c(3,15),
          kfile,cafile,dfile,dlabels)
plotcorr(X)
```

---

redux *The* redux *class*

Description

An object class that is used throughout Ar-Ar_Redux

Details

A list with the following items:

labels: a vector of strings denoting the names of the runs
num: a vector of strings denoting the numerator isotopes
den: a vector of strings denoting the denominator isotopes
intercepts: a vector of logratio intercepts or values
covmat: the covariance matrix of intercepts
irr: a vector of strings denoting the irradiation runs
pos: a vector of integers with the positions in the irradiation stack
thedate: a vector containing the acquisition dates and times
nlr: a vector with the number of logratios per run
param: a list of global parameters

See Also

param

`redux2isoplotr` *Export* `ArArRedux` *data to* `IsoplotR`

Description

Creates a data object compatible with the `IsoplotR` package

Usage

```
redux2isoplotr(x, irr, fract = NULL, ca = NULL, k = NULL, format = 1,
  file = NULL)
```

Arguments

`x`	an object of class `redux`
`irr`	the irradiation schedule
`fract`	list with air shot data (one item per denominator isotope)
`ca`	an object of class `logratios` with Ca-interference data (not necessary if interferences are included in X)
`k`	an object of class `logratios` with K-interference data (not necessary if interferences are included in X)
`format`	input format for `IsoplotR`. I.e. one of 1: 39/40, s[39/40], 36/40, s[36/40], 39/36, s[39/36] (other formats will be added later)
`file`	optional (`.csv`) file name to write the output to.

Value

an object of class `ArAr`, i.e. a table with the following columns: `'Ar4036'`, `'errAr4036'`, `'Ar3936'`, `'errAr3936'`, `'Ar4039'`, and `'errAr4039'`

Examples

```
data(Melbourne)
print(redux2isoplotr(Melbourne$X,Melbourne$irr))
```

results *The* results *class*

Description

A list with the following items:

Details

labels: a vector of strings denoting the names of the runs
intercepts: a vector of ages
covmat: the covariance matrix of intercepts
thedate: a vector containing the acquisition dates and times

subset.timeresolved *Select a subset of some data*

Description

Extracts those intercepts, covariances etc. that match a given list of indices or labels.

Usage

```
## S3 method for class 'timeresolved'
subset(x, i = NULL, labels = NULL, invert = FALSE,
  include.J = FALSE, ...)

## S3 method for class 'logratios'
subset(x, i = NULL, labels = NULL, invert = FALSE,
  include.J = FALSE, ...)

## S3 method for class 'redux'
subset(x, i = NULL, labels = NULL, invert = FALSE,
  include.J = FALSE, ...)

## S3 method for class 'results'
subset(x, i = NULL, labels = NULL, invert = FALSE, ...)
```

Arguments

x	an object of class timeresolved, logratios, redux or results
i	a vector with indices of the selected runs
labels	a string or a vector of strings with sample names

invert	boolean flag indicating whether the selection should be inverted, i.e. whether the selected indices or labels should be removed rather than retained
include.J	if TRUE, automatically adds the irradiation monitors to the selection
...	other arguments

Value

an object of the same class as x

Examples

```
data(Melbourne)
ages <- process(Melbourne$X,Melbourne$irr,Melbourne$fract)
MD <- subset(ages,labels=c("MD2-1","MD2-2","MD2-3","MD2-4","MD2-5"))
plotcorr(MD)
```

---

summary.results *Summary table*

Description

Plots the ages and their standard errors

Usage

```
## S3 method for class 'results'
summary(object, ...)
```

Arguments

object	an objct of class results
...	no other arguments

Examples

```
data(Melbourne)
ages <- process(Melbourne$X,Melbourne$irr,Melbourne$fract)
summary(ages)[1:5,]
```

timeresolved *The* timeresolved *class*

Description

An object class containing time resolved multi-collector mass spectrometry data

Details

A list with the following items:

masses: a vector of strings denoting the isotopes monitored in each run
irr: a vector of strings denoting the irradiation runs
pos: a vector of integers with the positions in the irradiation stack
thedate: a vector containing the acquisition dates and times
d: a data table
thetime: a matrix with the measurement times

See Also

loaddata

weightedmean *Calculate the weighted mean age*

Description

Computes the error weighted mean and MSWD of some samples taking into covariances.

Usage

```
weightedmean(ages, prefix = NULL)
```

Arguments

ages	an object of class results
prefix	is either a string with the prefix of the samples that need to be averaged, or a vector of sample names.

Value

a list with items:

avgt: the weighted mean age
err: the standard error of avgt
MSWD: the Mean Square of the Weighted Deviates

Examples

```
data(Melbourne)
ages <- process(Melbourne$X,Melbourne$irr,Melbourne$fract)
weightedmean(ages,"MD2-")
```

Index