Package 'sandbox'

September 12, 2024

Type Package

Title Probabilistic Numerical Modelling of Sediment Properties

Version 0.2.1

Date 2022-02-25

Author Michael Dietze [aut, cre] (<https://orcid.org/0000-0001-6063-1726>), Sebastian Kreutzer [aut] (<https://orcid.org/0000-0002-0734-2199>)

Maintainer Michael Dietze <mdietze@gfz-potsdam.de>

Description A flexible framework for definition and application of time/depth-based rules for sets of parameters for single grains that can be used to create artificial sediment profiles. Such profiles can be used for virtual sample preparation and synthetic, for instance, luminescence measurements.

License GPL-3

BugReports https://github.com/coffeemuggler/sandbox/issues

Depends R (>= 4.0)

LazyData TRUE

Imports methods, RLumModel (>= 0.2.9), parallel

Suggests EMMAgeo (>= 0.9.7), Luminescence (>= 0.9.15), testthat (>= 3.0.2),

Encoding UTF-8

Language en-GB

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-02-25 08:10:02 UTC

Contents

`sandbox-package` *Probabilistic Numerical Modelling of Sediment Properties*

Description

Flexible framework for definition and application of time/depth-based rules for sets of parameters for single grains that can be used to create synthetic samples, used for synthetic preparation and synthetic measurements.

Author(s)

Michael Dietze (GFZ Potsdam, Germany), Sebastian Kreutzer (Geography & Earth Sciences, Aberystwyth University, United Kingdom)

`add_Population` *Add a Population to a Rule Book*

Description

The function adds a further population element to all rules or a rule book.

Usage

```
add_Population(book, populations = 1)
```

Arguments

`book`	character value, name of the rule book to be modified.
`populations`	numeric value, number of additional populations to create.

Value

A list object with all rules for a model run.

Author(s)

Michael Dietze, GFZ Potsdam (Germany)

Examples

```
## create simple true age-depth-relationship
book_1 <- get_RuleBook()

book_2 <- add_Population(
 book = book_1,
 populations = 1)
```

`add_Rule` *Add a Rule to a Rule Book*

Description

The function adds a new rule to an existing rule book. The specified rule will be appended to the rule book.

Usage

```
add_Rule(book, name, group, type, populations = 1)
```

Arguments

`book`	character value, name of the rule book to be modified.
`name`	character value, name of the rule to be added.
`group`	character value, group to which the rule belongs. One out of `"general"` (covering the sediment section properties) and `"specific"` (relevant for a single grain).
`type`	character value, generic type of the rule. One out of `"exact"` (defined by exact value, changing with depth), `"normal"` (normal distribution, defined by mean and standard deviation, changing with depth), `"uniform"` (defined by minimum and maximum values, changing with depth) and `"gamma"` (gamma distribution, defined by shape and scale parameter and constant offset, all changing with depth)
`populations`	numeric value, number of populations to create. The number of populations to add should match the existing number of populations.

Value

A list object with all rules for a model run.

Author(s)

Michael Dietze, GFZ Potsdam (Germany), Sebastian Kreutzer, Geography & Earth Sciences, Aberystwyth University (United Kingdom)

Examples

```
## create simple true age-depth-relationship
book_1 <- get_RuleBook()

book_2 <- add_Rule(
 book = book_1,
 name = "extrarule",
 group = "general",
 type = "normal",
 populations = 1)
```

`convert_units` *Convert between phi units and micrometers*

Description

The function converts values from the phi-scale (Krumbein 1934, 1938) to the micrometer-scale and vice versa.

Usage

```
convert_units(phi, mu)
```

Arguments

`phi`	numeric vector, grain-size class values in phi to be converted
`mu`	numeric vector, grain-size class values in micrometres to be converted

Details

$$\phi = -log2(D/D_0)$$

with D the diameter in µm and D_0 the reference diameter. Herer 1000 µm.

Value

numeric vector, converted grain-size class values

Author(s)

Michael Dietze, GFZ Potsdam (Germany)

References

Krumbein, W.C., 1938. Size frequency distributions of sediments and the normal phi curve. Journal of Sedimentary Research 8, 84–90. doi: 10.1306/D42690082B2611D78648000102C1865D

Krumbein, W.C., 1934. Size frequency distributions of sediments. Journal of Sedimentary Research 4, 65–77. doi: 10.1306/D4268EB92B2611D78648000102C1865D

Examples

```
## load example data set
## generate phi-values
phi <- -2:5

## convert and show phi to mu
mu  <- convert_units(phi = phi)
mu

## convert and show mu to phi
convert_units(mu = mu)
```

---

get_RuleBook *Get One of a Series of Predefined Rule Books for a Model Run.*

Description

The function returns a pre-built model rule book, i.e., a combination of model parameters and rules.

Usage

```
get_RuleBook(book = "empty", osl = NULL)
```

Arguments

book	character value, name of the rule book to be generated. One out of "empty", default is "empty".
osl	character value, optional keyword for an OSL (optical stimulated luminescence) model of choice. Must be one of the available models from the R package RLumModel-package. See details for full list of available models.

Details

It is possible to generate OSL-tailored rule books. For this, the argument osl must be provided with a keyword defining one of the OSL models from the R package 'RLumModel': "Bailey2001", "Bailey2004", "Pagonis2008", "Pagonis2007", "Bailey2002" and "Friedrich2017". The model parameters will be appended to the rule book entries and defined by mean and standard deviation.

Value

A list object with all rules for a model run.

Author(s)

Michael Dietze, GFZ Potsdam (Germany), Sebastian Kreutzer, Geography & Earth Sciences, Aberystwyth University (United Kingdom)

Examples

```
## create simple true age-depth-relationship
book_flat <- get_RuleBook(book = "empty")
```

make_Sample *Create a Virtual Sample.*

Description

The function generates many virtual sediment grains based on the specified sample geometry and depth, using the information from a rule book.

Usage

```
make_Sample(
  book,
  depth,
  geometry = "cuboid",
  radius,
  height,
  width,
  length,
  slice = TRUE,
  force = FALSE,
  n_cores = max(1, parallel::detectCores() - 2)
)
```

Arguments

book	list object, initially produced by get_RuleBook
depth	numeric scalar, depth of the sample centre (m).
geometry	character scalar, keyword defining the geometry of the sample. One out of "cuboid" and "cylinder", default is "cuboid".
radius	numeric scalar, radius of the cylinder (m).
height	numeric scalar, height of the cuboid (m).
width	numeric scalar, width of the cuboid (m).
length	numeric scalar, length of the cuboid or cylinder (m).
slice	logical scalar, option to sample in repeated slices of 10^6 grains until the required sample size is reached. Useful to avoid memory issues for large numbers of grains per sample volume.
force	logical scalar, option to override the default maximum number of 10^7 grains per sample, set to avoid memory problems of the computer.
n_cores	integer (*optional*) set the number of cores used for the parallel processing

Value

A list object.

Author(s)

Michael Dietze, GFZ Potsdam (Germany)

Examples

```
set.seed(12234)
sample_01 <- make_Sample(
  book = get_RuleBook(),
  depth = 1,
  geometry = "cuboid",
  n_cores = 1,
  height = 0.001,
  width = 0.001,
  length = 0.001)
```

`measure_SAR_OSL` *Measure an aliquot with the CW SAR OSL protocol*

Description

The function models the time-dependent photon counts of an aliquot according to the specified CW SAR OSL (continuous wave, single aliquot regenerative dose protocol for optically stimulated luminescence) sequence and parameters. The modelling is done for each component and photon count curves are summed to return an Luminescence::RLum.Analysis object as equivalent of importing a real measurement data set to the R-package `Luminescence-package`.

The function uses the package RLumModel-package to perform the simulation of the photon count curves.

Usage

```
measure_SAR_OSL(aliquot, sequence, dose_rate = 0.1)
```

Arguments

`aliquot`	data.frame or a list of it, a set of grains that are assigned to an aliquot (sample subset used for measurement), i.e., the result of prepare_Aliquot.
`sequence`	list, definition of the SAR protocol.
`dose_rate`	numeric value, Dose rate of the luminescence reader, in Gy/s.

Value

Luminescence::RLum.Analysis object. Equivalent of the import result for a real world measurement file. This object can be evaluated by functions of the package `Luminescence-package`.

Author(s)

Michael Dietze, GFZ Potsdam (Germany), Sebastian Kreutzer, Geography & Earth Sciences, Aberystwyth University (United Kingdom)

Examples

```
## Not run:

## load example data set
data(sample_osl_aliquots, envir = environment())

sequence <- list(
  RegDose = c(0, 1, 2, 5, 10, 0, 1),
  TestDose = 2,
  PH = 220,
  CH = 200,
  OSL_temp = 125,
  OSL_duration = 70)

## reduce number of
## grains to two
sample_osl_aliquots$aliquot_1 <-
sample_osl_aliquots$aliquot_1[1:2,]

## or measure all aliquots in a row
sar_all <- measure_SAR_OSL(
 aliquot = sample_osl_aliquots,
 sequence = sequence,
 dose_rate = 0.1)


## End(Not run)
```

`prepare_Aliquot` *Prepare Aliquots from Sample Dataset*

Description

The function consecutively fills aliquots (i.e., subsamples distributed on round carrier discs) with grains from an input sample. Remaining grains that are not enough to fill a further aliquot are discarded.

Usage

```
prepare_Aliquot(sample, diameter, density = 0.65)
```

Arguments

`sample`	data.frame, sample object to be distributed to aliquots.
`diameter`	numeric value, diameter of the aliquot sample carriers in mm.
`density`	numeric value, packing density of the grains on the sample carrier. Default is `0.65`. The packing density is unitless.

Value

list of data.frame objects with grains organised as aliquots, i.e. list elements.

Author(s)

Michael Dietze, GFZ Potsdam (Germany), Sebastian Kreutzer, Geography & Earth Sciences, Aberystwyth University (United Kingdom)

Examples

```
## load example data set
data(sample, envir = environment())

A <- prepare_Aliquot(
 sample = sample,
 diameter = 0.1)

B <- prepare_Aliquot(
 sample = sample,
 diameter = 1,
 density = 0.6)
```

---

prepare_Sieving *Sieve a Sample*

Description

The function removes grains that are not within the provided sieve interval.

Usage

```
prepare_Sieving(sample, interval)
```

Arguments

sample	data.frame sample object to be sieved.
interval	numeric vector, sieve interval, in phi units.

Value

data.frame with grains that are within the sieve interval.

Author(s)

Michael Dietze, GFZ Potsdam (Germany)

Examples

```
## load example data set
data(sample, envir = environment())

## sieve sample (in phi units)
sample_sieved <- prepare_Sieving(
  sample = sample,
  interval = c(5, 6))
```

```
## plot results
plot(density(
  x = sample$grainsize,
  from = -1,
  to = 11))
lines(density(
  x = sample_sieved$grainsize,
  from = -1,
  to = 11),
  col = 2)
```

`prepare_Subsample` *Prepare Subsamples from a Sample Dataset*

Description

The function splits the master sample in a set of subsamples. The step can be done by creating equally large subsamples in terms of contained grains (parameter `number`), by volume (parameter `volume`) or by weight (parameter `weight`).

Usage

```
prepare_Subsample(sample, number, volume, weight)
```

Arguments

`sample`	data.frame, sample object to be distributed to aliquots.
`number`	numeric value, number of evenly large subsamples to be created
`volume`	numeric value, volume of subsamples. Remainder of the master sample that is too small for the last subsample is removed. Volume must be given in m^3 and takes packing density of the sample into account.
`weight`	numeric value, weight of the subsamples. Remainder of the master sample that is too small for the last subsample is removed. Weight is calculated based on density of each grain. Weight must be given in kg.

Value

list object with grains organised as aliquots, i.e. list elements.

Author(s)

Michael Dietze, GFZ Postdam (Germany)

Examples

```
## load example data set
data(sample, envir = environment())

## create 10 subsamples
prepare_Subsample(sample, 10)
```

sample *Example Grain Size Data*

Description

Example data set of a virtual loess-like sample.

Format

The format is: 'data.frame': 1000 obs. of 12 variables: $ ID : int 33107 33108 33109 33110 33111 33112 ... $ depth : num 5 5 5 5 5 ... $ population : num 3 1 3 2 1 3 1 3 3 3 ... $ age : num 25711 25710 25712 25709 25710 ... $ dose_rate : num 7.163 -1.083 -0.929 3.541 5.732 ... $ water_content : num 13.29 10.99 3.65 8.98 3.29 ... $ population : num 0.3 0.586 0.3 0.114 0.586 ... $ grainsize : num 4.01 6.22 5.16 5.47 5.57 ... $ density : num 1.92 1.91 1.9 1.88 1.9 ... $ packing : num 0.708 0.702 0.698 0.702 0.688 ... $ photon_equivalent: num 1.017 0.993 1.005 1 0.995 ... $ predose : num 2020 3106 1983 191 2387 ...

Details

The sample was created using the rule book book_1, a depth of 5 m and a cuboid sample geometry with 2 mm edge length.

Examples

```
## load example data set
data(sample, envir = environment())

## plot grain-size distribution
plot(density(sample$grainsize))
```

sample_osl_aliquots *Aliquots Prepared to Measured Virtually*

Description

Example data of virtually prepared aliquots ready to be measured

Format

The format is: 'data.frame': 2 obs. of 65 variables: ..$ grains : num [1:2] 1 2 ..$ d_sample : num [1:2] 2 2 ..$ population : num [1:2] 1 1 ..$ age : num [1:2] 1574 1578 ..$ population : num [1:2] 2 2 ..$ grainsize : num [1:2] 2.52 2.48 ..$ packing : num [1:2] 1.32 4.82 ..$ density : num [1:2] 3.24 2.13 ..$ osl_doserate: num [1:2] 0.00875 0.0046 ..$ osl_N1 : num [1:2] 1.5e+07 1.5e+07 ..$ osl_N2 : num [1:2] 1e+07 1e+07 ..$ osl_N3 : num [1:2] 1e+09 1e+09 ..$ osl_N4 : num [1:2] 2.5e+08 2.5e+08 ..$ osl_N5 : num [1:2] 5e+10 5e+10 ..$ osl_N6 : num [1:2] 3e+08 3e+08 ..$ osl_N7 : num [1:2] 1e+10 1e+10 ..$ osl_N8 : num [1:2] 5e+09 5e+09 ..$ osl_N9 : num [1:2] 1e+11 1e+11 ..$ osl_E1 : num [1:2] 0.97 0.97 ..$ osl_E2 : num [1:2] 1.55 1.55 ..$ osl_E3 : num [1:2] 1.7 1.7 ..$ osl_E4 : num [1:2] 1.72 1.72 ..$ osl_E5 : num [1:2] 2 2 ..$ osl_E6 : num [1:2] 1.43 1.43 ..$ osl_E7 : num [1:2] 1.75 1.75 ..$ osl_E8 : num [1:2] 5 5 ..$ osl_E9 : num [1:2] 5 5 ..$ osl_s1 : num [1:2]

5e+12 5e+12 ..$ osl_s2 : num [1:2] 5e+14 5e+14 ..$ osl_s3 : num [1:2] 5e+13 5e+13 ..$ osl_s4 : num [1:2] 5e+14 5e+14 ..$ osl_s5 : num [1:2] 1e+10 1e+10 ..$ osl_s6 : num [1:2] 5e+13 5e+13 ..$ osl_s7 : num [1:2] 5e+14 5e+14 ..$ osl_s8 : num [1:2] 1e+13 1e+13 ..$ osl_s9 : num [1:2] 1e+13 1e+13 ..$ osl_A1 : num [1:2] 1e-08 1e-08 ..$ osl_A2 : num [1:2] 1e-08 1e-08 ..$ osl_A3 : num [1:2] 1e-09 1e-09 ..$ osl_A4 : num [1:2] 5e-10 5e-10 ..$ osl_A5 : num [1:2] 1e-10 1e-10 ..$ osl_A6 : num [1:2] 5e-07 5e-07 ..$ osl_A7 : num [1:2] 1e-09 1e-09 ..$ osl_A8 : num [1:2] 1e-10 1e-10 ..$ osl_A9 : num [1:2] 1e-09 1e-09 ..$ osl_B1 : num [1:2] 0 0 ..$ osl_B2 : num [1:2] 0 0 ..$ osl_B3 : num [1:2] 0 0 ..$ osl_B4 : num [1:2] 0 0 ..$ osl_B5 : num [1:2] 0 0 ..$ osl_B6 : num [1:2] 5e-09 5e-09 ..$ osl_B7 : num [1:2] 5e-10 5e-10 ..$ osl_B8 : num [1:2] 1e-10 1e-10 ..$ osl_B9 : num [1:2] 1e-10 1e-10 ..$ osl_Th1 : num [1:2] 0.75 0.75 ..$ osl_Th2 : num [1:2] 0 0 ..$ osl_Th3 : num [1:2] 6 6 ..$ osl_Th4 : num [1:2] 4.5 4.5 ..$ osl_Th5 : num [1:2] 0 0 ..$ osl_E_th1 : num [1:2] 0.1 0.1 ..$ osl_E_th2 : num [1:2] 0 0 ..$ osl_E_th3 : num [1:2] 0.1 0.1 ..$ osl_E_th4 : num [1:2] 0.13 0.13 ..$ osl_E_th5 : num [1:2] 0 0 ..$ osl_R : num [1:2] 5e+07 5e+07

Examples

```
## load example data set
data(sample_osl_aliquots, envir = environment())

## plot grain-size distribution
plot(density(sample_osl_aliquots[[1]]$age))
```

`set_Parameter` *Set Profile- and Grain-Specific Model Parameters.*

Description

The function defines one model parameter used to generate a set of virtual grains. A parameter is defined in a probabilistic way, as parametric distribution function. Each parameter of the distribution function can be changed through time using set_Rule.

Usage

```
set_Parameter(book, parameter, type)
```

Arguments

`book`	list object, rule book to be edited.
`parameter`	character scalar, keyword defining the parameter to be defined. Some parameters can be described by more than one function, see details.
`type`	character scalar, keyword defining the distribution function used to describe the parameter. See details for available keywords, default is `"exact"`.

Details

The following parameter types are available:

- `exact`: parameter does not vary at all. No additional parameters needed except for vector `value`, defining the constant values for corresponding depths.
- `uniform`: parameter varies following a uniform distribution. The following additional parameter vectors are required: `min` (minimum) and `max` (maximum)

- `normal`: parameter varies following a normal distribution, which is defined by mean and standard deviation
- `gamma`: parameter varies following a gamma distribution, defined by shape parameter, scale parameter) and offset (defining constant offset of values)

Value

A list object.

Author(s)

Michael Dietze, GFZ Potsdam (Germany)

Examples

```
## get empty rule book
book_1 <- get_RuleBook(book = "empty")

## set density from default "normal" to "exact"
book_2 <- set_Parameter(book = book_1,
                        parameter = "density",
                        type = "exact")

book_1$density$density_1$type
book_2$density$density_1$type
```

`set_Rule` *Set depth-dependent rule for model parameter.*

Description

The function defines how the specified model parameter varies with depth. The transfer function uses different interpolation functions to create a continuous representation of a parameter value with depth.

Usage

```
set_Rule(book, parameter, value, depth, type = "spline")
```

Arguments

`book`	list object, rule book to be edited.
`parameter`	character scalar, parameter name to be edited. Can also be the keyword for an OSL model. See details.
`value`	numeric list, specifying the parameter values at the corresponding depth points. If a parameter is defined by more than one argument (e.g., mean and standard deviation), all the relevant arguments must be defined for each corresponding depth as separate list element.
`depth`	numeric list, specifying the depths used for the interpolation. All elements must be of the same lengths as the corresponding data in `value`.
`type`	character scalar, interpolation method. One out of `spline`, default is `spline`.

Details

To assign standard OSL model parameters, one of the available keywords of the R package RLumModel-package can be used. The function will then set all rules of the rule book with the standard values associated with these models, and setting the standard deviation to zero. The keyword can be one out of `"Bailey2001"`, `"Bailey2004"`, `"Pagonis2008"`, `"Pagonis2007"`, `"Bailey2002"` and `"Friedrich2017"`. This will fill the rule book with the standard parameters independent of depth. Note that a dose rate (parameter name `osl_doserate`) needs to be set separately!

Value

A list object with all created formula objects.

Author(s)

Michael Dietze, GFZ Potsdam (Germany), Sebastian Kreutzer, Geography & Earth Sciences, Aberystwyth University (United Kingdom)

Examples

```
## create empty rule book
book_01 <- get_RuleBook()

## assign rule definitions to lists
depth <- list(c(0, 10))
age <- list(c(0, 1000))

## add age definition
book_01 <- set_Rule(
 book = book_01,
 parameter = "age",
 value = age,
 depth = depth)
```

Index