Package: ADMUR (via r-universe)

September 13, 2024

Version 1.0.3.9005

Date 2023-09-06

Title Ancient Demographic Modelling Using Radiocarbon

Maintainer Adrian Timpson <a.timpson@ucl.ac.uk>

Depends R (>= 3.5.0)

Imports graphics, stats, methods, mathjaxr

Suggests DEoptimR, knitr, rmarkdown, scales

Description Statistical tools to directly model underlying population dynamics using date datasets (radiocarbon and other). Various model structures can be tested including Continuous Piecewise Linear (CPL) models that are flexible to estimate any complex population dynamics, Uniform, Exponential, Gaussian, Cauchy, Sinusoidal, Logistic and Power law. Bayesian parameter estimates of population models. Taphonomic loss included optionally as a power function. Model comparison framework using BIC. Package also calibrates 14C samples, generates Summed Probability Distributions (SPD), and performs SPD simulation analysis to generate a Goodness-of-fit test for the best selected model. Details about the method can be found in Timpson A., Barberena R., Thomas M. G., Mendez C., Manning K. (2020) <https://royalsocietypublishing.org/doi/10.1098/rstb.2019.0723>.

License GPL-3

URL https://github.com/UCL/ADMUR

Encoding UTF-8

VignetteBuilder knitr

LazyData true

RdMacros mathjaxr

RoxygenNote 7.1.1

Repository https://ucl.r-universe.dev

RemoteUrl https://github.com/ucl/admur

RemoteRef HEAD

RemoteSha 838f661049b15cfcb514105bbf7ed900aa2e71ea

Contents

`ADMUR` *ADMUR R package*

Description

ADMUR: Ancient Demographic Modelling Using Radiocarbon

Tools to directly model and compare the underlying dynamics of prehistoric population change using radiocarbon date datasets.

Details

Package to model population dynamics from anthropogenic datasets of dates such as radiocarbon dates. Provides tools to perform and compare three modelling approaches:

1. Directly modelling the underlying population dynamics using common parameteric models, including uniform, gaussian, exponential, logistic, sinusoidal, cauchy, power, etc.

2. Directly modelling the underlying population dynamics using a Continuous Piecewise Linear (CPL) model framework, including estimating the dates and magnitude of demographic events (hinge points).

3. Directly modelling the underlying population dynamics using an independent timeseries.

4. Summed Probability Distribution (SPD) generation, and simulation testing. Permits the rejection of a null hypothesis.

Tools to estimate Maximum likelihood; Goodness of Fit tests; model comparison.

References

'Directly modelling population dynamics in the South American Arid Diagonal using 14C dates' by Adrian Timpson, Ramiro Barberena, Mark G. Thomas, César Méndez and Katie Manning, published in Philosophical Transactions of the Royal Society B, 2020. https://royalsocietypublishing.org/doi/10.1098/rstb.2019.0

'ADMUR: Ancient Demographic Modelling Using Radiocarbon. Adrian Timpson. University College London. Research Department of Genetics, Environment and Evolution (GEE), Darwin Building, Gower Street, London, WC1E 6BT. 2020 https://github.com/UCL/ADMUR

`bluhm2421` *Radiocarbon dataset from Bluhm and Surovell 2018*

Description

The 14C dataset used by Bluhm and Surovell in 'Validation of a global model of taphonomic bias using geologic radiocarbon', Quaternary Research 2018. Data provided from Supplementary materials originally comprised 2457 rows, but 36 rows (dates(have no mean or SD (only a minimum age) and have been excluded from this data frame. Furthermore, 25 rows (dates) have to SDs (+ve and -ve), which have been changed to a single SD for this dataset by taking the mean average.

Usage

```
bluhm2421
```

Format

A data frame comprising 2421 rows and 4 columns.

`bryson1848` *Radiocarbon dataset from Bryson et al. 2006*

Description

The 14C dataset used by Surovell et al. in 'Correcting temporal frequency distributions for taphonomic bias', Journal of Archaeological Science 2009. Data was originally published by Bryson et al. in 'A calibrated radiocarbon database of late Quaternary volcanic eruptions', Earth Discussions, 2006. Bryson et al. source data contains two data tables. The second comprises 173 rows of dates missing their standard deviations. Instead this data frame corresponds to Brysons first table comprising 1848 rows. However, these are often an amalgamation of several individual radiocarbon dates which cannot be reverse-engineered to obtain the individual dates. Therefore we assume each row represents a unique phase.

Usage

```
bryson1848
```

Format

A data frame comprising 1848 rows and 8 columns.

`checkData` *Checks a dataset for obvious clangers*

Description

Performs some rudimentary sanity checks on a radiocarbon (or other) dataset

Usage

```
checkData(data)
```

Arguments

`data`	A dataframe of 14C dates. Requires 'age' and 'sd', and at least one of 'site' and 'phase'. Optional 'datingType' to include '14C' and anything else.

Details

Performs some rudimentary checks on the radiocarbon dataset, ensuring structure is as required, ages and sds look vaguely sensible etc. This is no substitute for poor data hygiene, and the analyst should of course have a toolkit of many other checks, e.g., to avoid duplicate labcodes.

Value

NULL

Examples

```
checkData(SAAD)
```

`convertPars` *Converts parameters to x,y coordinates (date and pdf) that describe a model*

Description

Converts either a vector of parameters, or a matrix of many parameter sets to model x,y coordinates (date and pdf)

Usage

```
convertPars(pars, years, type, timeseries = NULL)
```

Arguments

`pars`	Either a numeric vector (one parameter set), or a matrix of several parameter sets (one set per row).
`years`	A vector of years.
`type`	Choose from the following currently available `models`. Composite models can be achieved using a vector of more than one type. For example, c('norm','power') will be a composite model, where the first two parameters are the mean and SD, the 3rd and 4th parameters determine the power distribution component, for example if modelling taphonomy.
`timeseries`	A data frame containing names x and y (date and pdf) must be provided as the timeseries, only if type is 'timeseries'. If 'type' is anything else, timeseries is not required (default = NULL).

Details

Converts model parameters into a timeseries. For example, a parameter search will yield either a single set of parameters, or a matrix with one parameter set per row (such as the 'res' value from `mcmc`). Either can be handed directly to this function. The structure of the output differs depending on if converting a vector or matrix.

Examples

```
#  a random 6-CPL parameter set
pars <- runif(11)
x <- convertPars( pars=pars, years=5500:7500, type='CPL')

#  a matrix of 5 random 6-CPL parameter sets
pars <- matrix( runif(11*5), 5, 11 )
x <- convertPars( pars=pars, years=5500:7500, type='CPL')

#  a random exponential parameter
pars <- runif(1, -0.01, 0.01)
x <- convertPars( pars=pars, years=5500:7500, type='exp')

#  a matrix of 5 random exponential parameter sets
pars <- matrix( runif(5, -0.01, 0.01), 5, 1 )
x <- convertPars( pars=pars, years=5500:7500, type='exp')

#  a random Gaussian parameter pair (mean, sd)
pars <- runif(2, c(6000,200), c(7000,1000))
x <- convertPars( pars=pars, years=5500:7500, type='norm')

# a combination model of a Gaussian (parameters = mean, sd),
# and a power model assumed to be a taphonomic effect (parameters = b,c).
pars <- runif(4, c(6000,200,0,-3), c(7000,1000,20000,0))
x <- convertPars( pars=pars, years=5500:7500, type=c('norm','power'))

# 5 combination models of a Gaussian (parameters = mean, sd),
# and a power model assumed to be a taphonomic effect (parameters = b,c).
pars <- t(matrix(runif(4*5, c(6000,200,0,-3), c(7000,1000,20000,0)),4,5))
x <- convertPars( pars=pars, years=5500:7500, type=c('norm','power'))

# a single random Cauchy parameter pair (location, scale)
pars <- runif(2, c(6000,200), c(7000,1000))
x <- convertPars( pars=pars, years=5500:7500, type='cauchy')

# a combination model of a Cauchy (parameters = location, scale),
# and a power model assumed to be a taphonomic effect (parameters = b,c).
pars <- runif(4, c(6000,200,0,-3), c(7000,1000,20000,0))
x <- convertPars( pars=pars, years=5500:7500, type=c('norm','power'))

# a single random logistic parameter pair (k, x0)
pars <- runif(2, c(0,6000), c(0.01,6500))
x <- convertPars( pars=pars, years=5500:7500, type='logistic')

# a combination model of a logistic (parameters = k, x0),
# and a power model assumed to be a taphonomic effect (parameters = b,c).
pars <- runif(4, c(0,6000,0,-3), c(0.01,6500,20000,0))
x <- convertPars( pars=pars, years=5500:7500, type=c('logistic', 'power'))

# a single random sinewave parameter set (f,p,r)
f <- 1/runif(1,200,1000)
p <- runif(1,0,2*pi)
```

```
r <- runif(1,0,1)
x <- convertPars( pars=c(f,p,r), years=5500:7500, type='sine')

# a combination model of a sinewave (parameters = f,p,r),
# and a power model assumed to be a taphonomic effect (parameters = b,c).

f <- 1/runif(1,200,1000)
p <- runif(1,0,2*pi)
r <- runif(1,0,1)
b <- runif(1,0,20000)
c <- runif(1,-3,0)
x <- convertPars( pars=c(f,p,r,b,c), years=5500:7500, type=c('sine','power'))

# although a uniform distribution has no parameters, a pdf can still be generated:
# pars must be set to NA
x <- convertPars(pars=NA, years=5500:7500, type='uniform')

# a uniform combined with a power model (assumed to be a taphonomic effect):
# the parameter for the uniform component must be set to NA
pars <- c(NA, runif(2, c(0,-3), c(20000,0)))
x <- convertPars(pars=pars, years=5500:7500, type=c('uniform','power'))
```

`CPLparsToHinges` *Converts CPL parameters (0 to 1) into hinge (x,y) coordinates (date and pdf) that describe a model*

Description

Converts either a vector of parameters, or a matrix of many parameter sets to CPL-model hinges (date and pdf coordinates)

Usage

```
CPLparsToHinges(pars, years)
```

Arguments

`pars`	Either a vector of one parameter set, or a matrix of several parameter sets (one set per row).
`years`	A vector of years.

Details

The CPL model requires pars to be of odd length, each between 0 and 1. A n-CPL model has $2n-1$ parameters ($n-1$ x-parameters and n y-parameters) that are mapped to $n+1$ PD coordinates (x,y pairs) using a modified Stick Breaking Dirichlet Process. The first and last x-coordinate are set as x_{min} and x_{max}, and the remaining internal x-coordinates are converted from their respective x-parameters using the Beta distribution CDF (where $\alpha = 1$ and $\beta =$ the number of pieces still to be broken). The y-parameters (between 0 and 1) are converted to y-coordinates in two steps. Firstly, they are mapped from the parameter range (0,1) to the coordinate range (0, ∞) using the formula $\frac{1}{(1-y)^2} - 1$, and the single remaining y-coordinate is set as $\frac{1}{(1-0.5)^2} - 1$. Secondly, they are normalised by the total area under the curve, calculated as the sum of the areas under all n pieces:

$$Area = \sum_{i=1}^{n} (\frac{y_i + y_{i+1}}{2})(x_{i+1} - x_i)$$

Examples

```
# convert a single random 6-CPL parameter set
x <- CPLparsToHinges(pars=runif(11), years=5500:7500)
```

`data1` *Toy radiocarbon dataset*

Description

Data frame comprising a randomly generated 14C dataset. The true data generation process is deliberately left to the imagination

Usage

```
data1
```

Format

A data frame comprising 100 rows and 4 columns.

`data2` *Toy radiocarbon dataset*

Description

Data frame comprising a randomly generated 14C dataset. The true data generation process is deliberately left to the imagination

Usage

```
data2
```

Format

A data frame comprising 100 rows and 4 columns.

`data3` *Toy radiocarbon dataset*

Description

Data frame comprising a randomly generated 14C dataset. The true data generation process is deliberately left to the imagination

Usage

```
data3
```

Format

A data frame comprising 100 rows and 4 columns.

`data4` *Toy radiocarbon dataset*

Description

Data frame comprising a randomly generated 14C dataset. The true data generation process is deliberately left to the imagination

Usage

```
data4
```

Format

A data frame comprising 100 rows and 4 columns.

estimateDataDomain *Estimates the calendar date domain of a 14C dataset*

Description

Estimates the approximate date range of a 14C dataset, in calendar time

Usage

```
estimateDataDomain(data, calcurve)
```

Arguments

`data`	A dataframe of 14C dates. Requires 'age' and 'sd', and at least one of 'site' and 'phase'. Optional 'datingType' to include '14C' and anything else.
`calcurve`	A calibration curve object. Choose from intcal20 (default), shcal20, intcal13 or shcal13.

Details

Since dates are derived from normal (or log normal) distributions, they have no absolute cut off point. However, in practice the tail of a PDF rapidly becomes vanishingly small, so a date range can be estimated using an arbitrarily large interval (0.000001 to 0.999999) of the cumulative distribution.

In practice however, the date range chosen to model should not be selected using this function, but carefully chosen to ensure the date range is fairly represented by the data set.

Therefore this function should only be used to select a date range to model if simulating a tiny sample size, to ensure the modelled range does not exceed the date domain of the data.

Value

Returns a vector of two calendar dates BP.

Examples

```
# a single date within the 14C range 5000 to 10000
data <- data.frame(
  age = round(runif(1,5000,10000)),
  sd = 3,
  datingType = '14C'
  )
estimateDataDomain(data, calcurve=intcal20)

# 50 dates within the 14C range 5000 to 10000
data <- data.frame(
  age = round(runif(50,5000,10000)),
  sd = rep(50,50),
  datingType = rep('14C',50)
  )
estimateDataDomain(data, calcurve=intcal20)
```

`getModelChoices` *Lists currently available models*

Description

Lists currently available models, the number of parameters, and a brief description

Usage

```
getModelChoices()
```

Details

All models are truncated, such that the total area between x_{min} and x_{max} (the date range of the argument 'years') equals 1.

- 'CPL' is a Continuous Piecewise Linear model. It must have an odd number of parameters, each with a value between 0 and 1. A n-CPL model has $2n-1$ parameters ($n-1$ x-parameters and n y-parameters) that are mapped to $n+1$ PD coordinates (x,y pairs) using a modified Stick Breaking Dirichlet Process. The first and last x-coordinate are set as x_{min} and x_{max}, and the remaining internal x-coordinates are converted from their respective x-parameters using the Beta distribution CDF (where $\alpha = 1$ and $\beta =$ the number of pieces still to be broken). The y-parameters (between 0 and 1) are converted to y-coordinates in two steps. Firstly, they are mapped from the parameter range (0,1) to the coordinate range (0, ∞) using the formula $\frac{1}{(1-y)^2} - 1$, and the single remaining y-coordinate is set as $\frac{1}{(1-0.5)^2} - 1$. Secondly, they are normalised by the total area under the curve, calculated as the sum of the areas under all n pieces:
$$Area = \sum_{i=1}^{n} (\frac{y_i + y_{i+1}}{2})(x_{i+1} - x_i)$$
- 'uniform' is a uniform model requiring no parameters. I.e. the argument pars must be NULL, and trivially the PDF is:
$$\frac{1}{x_{max} - x_{min}}$$
- 'norm' is a truncated Gaussian model. The two parameters are used as μ and σ in the formula for a truncated Normal distribution, the PDF of which is calculated in two steps. Firstly, the PDF of an ordinary Normal distribution is calculated. Secondly, it is normalised by the area within the date range.
- 'exp' is a truncated exponential model of the form $f(x) = ae^{rx}$ where $x =$ years. The single parameter is used as the rate exponent r which gives growth through time if $r > 0$, decline if $r < 0$, and constant if $r = 0$. The PDF is as follows. Note the a parameter cancels out:
$$\frac{-re^{-rx}}{e^{-rx_{max}} - e^{-rx_{min}}}$$

- ’logistic’ is a truncated logistic model. The two parameters are used as the rate k and centre x_0 where the PDF is:

$$\frac{k}{(e^{-k(x_0-x)}+1)\ln\left(\frac{e^{-k(x_0-x_{min})}+1}{e^{-k(x_0-x_{max})}+1}\right)}$$

- ’sine’ is a truncated sinewave model. The usual function to describe a sine wave is $f(x) = A\sin(2\pi fx + p)$, where A is the amplitude, f is the frequency (cycles per year), and p is the cycle position (in radians) at $x = 0$, and therefore oscillates above and below the x-axis.

  However, a sinusoidal PDF must by definition always be non-negative, which can conceptually be considered as a sine wave stacked on top of a uniform distribution with a height $A + k$, where $k >= 0$. Since the PDF is $f(x)$ divided by the area below the curve, A and k simplify to a single parameter r that determines the relative proportions of the uniform and sinusoidal components, such that:

  when $r = 0$ the amplitude of the sine wave component is zero, and the overall PDF is just a uniform distribution between min and max.

  when $r = 1$ the uniform component is zero, and the minima of the sine wave touches zero. This does not necessarily mean the PDF minimum equals zero, since a minimum point of the sine wave may not occur with PDF domain (truncated between min and max).

  Therefore the formula for the PDF is:

$$\frac{1+\sin(2\pi fx+p)-\ln(r)}{(x_{max}-x_{min})(1-\ln(r))+(\frac{1}{2\pi f})[\cos(2\pi fx_{min}-p)-\cos(2\pi fx_{max}-p)]}$$

  where $x =$ years, and x_{min} and x_{max} determine the truncated date range;

  f determines the numeric frequency (cycles per unit x);

  p determines the cycle position (in radians) at $x = 0$, and must be a numeric between 0 and 2π;

  r determines how flat the distribution is, and must be a numeric between 0 and 1.

- ’cauchy’ is a truncated Cauchy model. The two parameters are used as x_0 (location) and γ (scale) in the formula for a truncated Cauchy distribution. The PDF is as follows where $x =$ years:

$$\frac{1}{\gamma[1+(\frac{x-x_0}{\gamma})^2][\arctan(\frac{x_0-x_{min}}{\gamma})-\arctan(\frac{x_0-x_{max}}{\gamma})]}$$

- ’power’ is a truncated Power function model of the form $f(x) = a(b + x)^c$ where $x =$ years. The PDF is as follows. Note the a parameter cancels out:

$$\frac{(c+1)(b+x)^c}{(b+x_{max})^{(c+1)}-(b+x_{min})^{(c+1)}}$$

- ’timeseries’ is a custom model of any complexity, typically derived from an independent source. For example, a timeseries of the proportion of forest to grassland through time, derived from pollen data. This requires a single parameter r to scale the PDF.

Examples

```
getModelChoices()
```

`intcal13` *Northern hemisphere 2013 calibration curve*

Description

Northern hemisphere 2013 calibration curve

Usage

```
intcal13
```

Format

A data frame comprising 5141 rows and 3 columns: cal BP, 14C BP, +/- Error.

Source

Atmospheric data from Reimer et al (2013). Obtained from the raw intcal13.14c file downloaded from http://radiocarbon.webhost.uits.arizona.edu/node/19

References

Reimer PJ, Bard E, Bayliss A, Beck JW, Blackwell PG, Bronk Ramsey C, Buck CE, Cheng H, Edwards RL, Friedrich M, Grootes PM, Guilderson TP, Haflidason H, Hajdas I, Hatte C, Heaton TJ, Hogg AG, Hughen KA, Kaiser KF, Kromer B, Manning SW, Niu M, Reimer RW, Richards DA, Scott EM, Southon JR, Turney CSM, van der Plicht J. Radiocarbon 55(4). DOI: 10.2458/azu_js_rc.55.16947

`intcal20` *Northern hemisphere 2020 calibration curve*

Description

Northern hemisphere 2020 calibration curve

Usage

```
intcal20
```

Format

A data frame comprising 9501 rows and 3 columns: cal BP, 14C BP, +/- Error.

Source

Atmospheric data from Reimer et al (2020). Obtained from the raw intcal20.14c file downloaded from http://intcal.org/curves/intcal20.14c

References

Reimer P, Austin WEN, Bard E, Bayliss A, Blackwell PG, Bronk Ramsey C, Butzin M, Cheng H, Edwards RL, Friedrich M, Grootes PM, Guilderson TP, Hajdas I, Heaton TJ, Hogg AG, Hughen KA, Kromer B, Manning SW, Muscheler R, Palmer JG, Pearson C, van der Plicht J, Reimer RW, Richards DA, Scott EM, Southon JR, Turney CSM, Wacker L, Adolphi F, Büntgen U, Capano M, Fahrni S, Fogtmann-Schulz A, Friedrich R, Köhler P, Kudsk S, Miyake F, Olsen J, Reinig F, Sakamoto M, Sookdeo A, Talamo S. 2020. The IntCal20 Northern Hemisphere radiocarbon age calibration curve (0-55 cal kBP). Radiocarbon 62. doi: 10.1017/RDC.2020.41.

`loglik` *Calculates the log likelihood of a population model, given a calibrated date PD matrix*

Description

Calculates the log likelihood of a population model, given a calibrated date PD matrix

Usage

```
loglik(PD, model)
```

Arguments

`PD`	A data frame of Probability Distributions (PDs). Each column representing the PD of a calibrated observation or phase. Row names correspond to the calendar years BP. This data frame can be generated by `phaseCalibrator`
`model`	A data frame containing the columns 'pdf' representing a hypothesised population Probability Density Function; and 'year' corresponding to the calendar years BP.

Details

Row names of both PD and model arguments must exactly match, since the probability of each phase given the model is calculated numerically.

Value

Returns a single numeric log likelihood.

Examples

```
# calibrate a dataset comprising just two phases
data <- data.frame(age=c(5800, 5100),sd=c(40, 35),phase=c('p1', 'p2'), datingType='14C')
CalArray <- makeCalArray(shcal20, calrange = range(toy$year))
PD <- phaseCalibrator(data, CalArray)

# calculate toy model log likelihood
loglik(PD, toy)
```

`makeCalArray` *Makes a calibration curve probability array*

Description

Generates CalArray containing a 2D probability array of the calibration curve

Usage

```
makeCalArray(calcurve, calrange, inc=5)
```

Arguments

`calcurve`	A calibration curve object. Choose from intcal20 (default), shcal20, intcal13 or shcal13.
`calrange`	A vector of two calendar dates BP, giving the calendar range of CalArray. Can be in either order.
`inc`	Increments to interpolate calendar years. Default = 5

Details

Generates an array of probabilities mapping the calibration curve and its error ribbon.

Each column represents a Gaussian PDF constructed from the C14 date and error of the calibration curve (typically a column every 5 cal years).

Row names of CalArray are 14C dates, column names are cal dates.

This function is memory and time costly if used to construct the entire 50,000 year range of the calibration curve at a resolution of 1 cal years, therefore typically used only for a constrained date range, by specifying calrange.

This array only needs constructing once to generate a Summed Probability Distribution of any number of calibrated dates, allowing efficient downstream calibration.

Value

Returns a list including:

`probs`	a 2D probability array of the calibration curve
`calcurve`	the calibration curve as provided as an argument
`cal`	a numeric vector of calendar years
`inc`	the resolution of the array in calendar years

Examples

```
# generate a CalArray of the intcal20 curve covering 5500 calBP to 7000 calBP
x <- makeCalArray(intcal13, c(5500,7000), inc=1 )
```

---

mcmc *Makes a MCMC chain using the Metropolis-Hastings algorithm*

Description

Generates a single Markov Chain Monte Carlo using the Metropolis-Hastings algorithm

Usage

```
mcmc(PDarray,
      startPars,
      type,
      timeseries=NULL,
      N=30000,
      burn=2000,
      thin=5,
      jumps=0.02)
```

Arguments

PDarray	A Probability Density Array of calibrated dates, generated by phaseCalibrator .
startPars	A vector of parameter values for the algorithm to start at. Suggest using the parameters found from a Maximum Likelihood search. Must have an odd length with values between 0 and 1 for an n-CPL model, or length 1 values between -0.1 and 0.1 for an exponential model.
type	Choose from the following currently available models. Composite models can be achieved using a vector of more than one type. For example, c('norm','power') will be a composite model, where the first two parameters are the mean and SD, the 3rd and 4th parameters determine the power distribution component, for example if modelling taphonomy.
timeseries	Only if 'type' = 'timeseries', a data frame should be provided containing columns x and y that define the timeseries as year and value respectively.
N	The total number of proposals made, before the chain is reduced for burn-in and thinning.
burn	The number of proposals made at the beginning of the chain to be disregarded as burn-in.
thin	Specifies the proportion of proposals to be disregarded (after burn-in), such that the default 5 will only keep every 5th proposal.

`jumps`	Vector that determines the size of the random jump between the last parameters and the proposed parameters. A smaller value gives a smaller jump. Different jump values can be provided for each parameter. This can be tuned by observing how well mixed each parameter is in the chain.

Details

Generates a single MCMC chain using the Metropolis-Hastings algorithm, printing to screen progress every 1000 proposals. An acceptance ratio of around 0.4 to 0.5 should be sought by adapting the arguments 'burn' and 'jumps'. If the acceptance ratio is too low try reducing jump. A larger dataset will typically require smaller jumps.

Value

Returns a list including:

`res`	A 2D matrix of free parameter values (between 0 and 1) in the chain, after burn-in and thinning.
`all.pars`	A 2D matrix of free parameter values (between 0 and 1) in the chain of all N proposals.
`acceptance.ratio`	The proportion of proposals (after burn-in) that are accepted.

Examples

```
# randomly sample calendar dates from the toy model
set.seed(12345)
N <- 350
cal <- simulateCalendarDates(toy, N)

# Convert to 14C dates.
age <- uncalibrateCalendarDates(cal, shcal20)
data <- data.frame(age = age, sd = 50, phase = 1:N, datingType = '14C')


# Calibrate each phase, taking care to restrict to the modelled date range
CalArray <- makeCalArray(shcal20, calrange = range(toy$year), inc = 5)
PD <- phaseCalibrator(data, CalArray, remove.external = TRUE)

# Run MCMC for a 3-CPL model (5 parameters)
chain <- mcmc(PDarray = PD,
 startPars = rep(0.5,5),
 type='CPL',
 jumps = 0.02)

# Run MCMC for a 3-CPL model with taphonomy (5 + 2 parameters)
chain <- mcmc(PDarray = PD,
 startPars = c(rep(0.5,5),10000,-1.5),
 type=c('CPL', 'power'),
 jumps = 0.02)
```

`objectiveFunction` *Objective function to be minimised in a search. Returns the negative log likelihood*

Description

Calculates the negative log likelihood of a model given a calibrated date matrix

Usage

```
objectiveFunction(pars, PDarray, type, timeseries=NULL)
```

Arguments

`pars`	A numeric vector of parameters.
`PDarray`	A data frame typically generated by `phaseCalibrator`, such that each column represents the PD of each calibrated date (or phase), and row names are the corresponding years.
`type`	Choose from the following currently available `models`. Composite models can be achieved using a vector of more than one type. For example, c('norm','power') will be a composite model, where the first two parameters are the mean and SD, the 3rd and 4th parameters determine the power distribution component, for example if modelling taphonomy.
`timeseries`	Only if 'type' = 'timeseries', a data frame should be provided containing columns x and y that define the timeseries as year and value respectively.

Details

If type is 'CPL', pars must be an odd length each between 0 and 1 since parameters correspond to: (y,x,...y). If type is 'exp', pars must be a single positive or negative numeric (the exponential rate can be growth or decay). Typically this parameter should be close to zero (-0.1 to 0.1) to avoid numbers beyond floating point limits) If type is 'norm', pars must have length 2 (mean and sd) If type is 'uniform', then pars must be NA.

Value

Returns a single value, the negative log likelihood.

Examples

```
# generate a PD array from a handful of dates
data <- subset(SAAD, site %in% c('Carrizal','Pacopampa'))
CalArray <- makeCalArray(shcal13, calrange = c(2000,6000))
PD <- phaseCalibrator(data, CalArray)

# the negative log likelihood given some random parameters for a 3-CPL model
pars <- runif(5)
```

```
objectiveFunction(pars, PD, type='CPL')

# the negative log likelihood given a random exponential model
pars <- runif(1, -0.01, 0.01)
objectiveFunction(pars, PD, type='exp')

# the negative log likelihood given a random Gaussian model
pars <- c(runif(1, 2000, 6000), runif(1, 100, 1000))
objectiveFunction(pars, PD, type='norm')

# the negative log likelihood given a uniform model
objectiveFunction(pars=NA, PD, type='uniform')

# the negative log likelihood given a uniform model with taphonomy
pars <- c(NA, runif(1, 0, 20000), runif(1, -3, 0))
objectiveFunction(pars=pars, PD, type=c('uniform','power'))
```

phaseCalibrator — *Generates an SPD for each phase in a dataset*

Description

Generates a data frame of SPDs, one for each phase

Usage

```
phaseCalibrator(data, CalArray, width = 200, remove.external = FALSE)
```

Arguments

data	A dataframe of 14C dates. Requires 'age' and 'sd', and at least one of 'site' and 'phase'. Optional 'datingType' comprising '14C' and/or anything else.
CalArray	A 2D probability array of the calibration curve generated by makeCalArray containing row names and column names.
width	A timespan in C14 years used to automatically bin dates if they have not been phased, i.e., 'phase' is missing from the data. Default = 200.
remove.external	Default FALSE retains the SPDs of all phases, even if some have little or no probability mass inside the date range. If TRUE, those phases with the majority of their probability mass outside the date range are removed.

Details

Generates an SPD for each phase using summedCalibrator with the default normalisation = 'standard'. Returns a data frame of probabilities, with phase names assigned to column names, and calendar years assigned to row names.

As a minimum requirement the data must include either 'phase' or 'site'. If 'phase' is unavailable the function will automatically use 'site' to bin the dates into site-phases. This binning is achieved with a crude algorithm that assigns a date to a bin if it is within 200 years of any other date in that bin, based on the uncalibrated C14 mean date (non-14C dates are also mapped to 14C time for the purpose of this binning). The need to bin dates into phases is an important step in modelling population dynamics to adjust for the data ascertainment bias of some archaeological finds having more dates by virtue of a larger research interest/budget. This binning algorithm provides a simple and useful solution to handling large datasets that have not been phased, but is not an alternative to an OxCal phase model if the objective is to directly estimate phase boundary dates at a specific site.

Optionally 'datingType' can be provided in the data. Only '14C' will be calibrated in the usual way, anything else is assumed to be provided in calendar time. If 'datingType' is not provided, all dates are assumed to be 14C.

Each column of the output data frame is a vector of probabilities representing the SPD of a phase. However, if the date range used to generate CalArray does not encompass the entire dataset, some phases will have PD outside the date range, giving a SPD area < 1. This avoids deleterious edge effects.

If using the output data frame to search for a population model, it is crucial to exclude dates outside (or mostly outside) the date range. This is achieved with remove.external = TRUE.

Value

Returns a data frame of probabilities. Each column provides the SPD of a phase. Column names are the phase names, row names are the calendar years.

Examples

```
CalArray <- makeCalArray(intcal20, calrange = c(9000,11000), inc = 5 )

# minimum data requirement includes 'mean' and 'sd' and either 'site' or 'phase'
data <- data.frame(age = c(8350,8500,8900,9200),
  sd = c(50,50,50,50),
  site = c('field','field','field','garden'))
x <- phaseCalibrator(data, CalArray)

# notice three phases were automatically generated, each with a total SPD area = 1
colSums(x)*5

# in contrast, three dates are specified as coming from the same phase,
# and the 'garden.1' phase is partly outside the date range
data <- data.frame(age = c(8350,8500,8900,9480),
  sd = c(50,50,50,50),
  phase = c('field.1','field.1','field.1','garden.1'))
x <- phaseCalibrator(data, CalArray)
colSums(x)*5
```

`plotCalArray` *Plots a calibration curve probability array*

Description

Generates a basic image plot of the calibration curve

Usage

```
plotCalArray(CalArray)
```

Arguments

`CalArray`	A 2D probability array of the calibration curve generated by `makeCalArray`, containing row names and column names.

Details

Plots CalArray, a 2D probability array of the calibration curve.

Time costly if CalArray comprises the entire 50,000 year range of the calibration curve.

Examples

```
# generate a CalArray of the intcal20 curve covering 5500 calBP to 6000 calBP
x <- makeCalArray( calcurve = intcal20, calrange = c(5500,6000), inc = 1 )
plotCalArray(x)
```

`plotPD` *Plots a calibrated PD of a single date, or SPD of multiple dates, or multiple SPDs*

Description

Generates a basic plot of the Probability Distribution of calibrated date, or Summed Probability Distribution of multiple calibrated dates, or multiple SPDs such as a data frame of phases

Usage

```
plotPD(x)
```

Arguments

`x`	A data frame comprising row names of calendar years and optional column names. For example a one-column dataframe generated by `summedCalibrator` or a multi-column dataframe generated by `phaseCalibrator`

Details

Presents the probability density as a grey polygon.

Examples

```
data <- data.frame(age=c(9144),sd=c(151))
CalArray <- makeCalArray(intcal20,calrange=c(8000,13000))
cal <- summedCalibrator(data, CalArray)
plotPD(cal)
```

---

plotSimulationSummary *Plots a summary of the SPD simulation test*

Description

Plots the SPD and confidence intervals of simulated SPDs, including regions outside the CI, the model, and 200 yr rolling mean

Usage

```
plotSimulationSummary(summary, title=NULL, legend.x=NULL, legend.y=NULL)
```

Arguments

`summary`	A list of various objects generated by `SPDsimulationTest`
`title`	A string title for the plot. If NULL a summary is automatically generated. If no title is preferred, use title = ”.
`legend.x`	The x coordinate for the figure legend.
`legend.y`	The y coordinate for the figure legend.

Details

Default NULL for legend.x and legend.y will automatically add a legend, which may not be ideally placed to avoid overlapping other components of the plot. To remove the legend, simply place well outside the boundary.

Examples

```

summary <- SPDsimulationTest(data=SAAD,
 calcurve=shcal20,
 calrange=c(2500,14000),
 pars=-0.0001674152,
 type='exp')

plotSimulationSummary(summary)
```

relativeRate	*Calculates the relative growth (or decline) rate per generation*

Description

Calculates the generational growth/decline rate for a linear piece of a CPL model, or between any two x,y coordinate pairs

Usage

```
relativeRate(x, y, generation = 25, N = 1000)
```

Arguments

`x`	A numeric vector of length 2, giving the start and end date of linear piece, or a 2 column matrix such that each row is a start and end pair.
`y`	The corresponding y values such as PDs, or population size (numeric vector of length 2), or a matrix such that each row is a start and end pair.
`generation`	Years per generation. Default = 25.
`N`	Number of sections to average the growth rate across.

Details

The 'relative rate' (growth or decline) of a straight line between two x,y coordinate pairs is the expected generational growth rate across this line. It is calculated always relative to the larger y-value, providing a symmetric measure. E.g., the absolute percentage changes from 80 to 100 to 80 are calculated as +20 The expected rate is the mean average of the conventional rates for N equal sections of the line, as N approaches infinity.

Value

Returns a numeric vector of values between 0 and 100 representing a 'relative percentage rate per generation'. Negative values indicate a decline through time, positive indicate growth.

Examples

```
x <- c(5600,5500)
y <- c(75,80)

# conventional growth/decline rate per 25 yr generation
100 * exp(log(y[2]/y[1])/((x[1]-x[2])/25)) - 100

# relative growth/decline rate per 25 yr generation
relativeRate(x,y)

x <- c(5600,5500)
y <- c(480,75)
```

```
# conventional growth/decline rate per 25 yr generation
100 * exp(log(y[2]/y[1])/((x[1]-x[2])/25)) - 100

# relative growth/decline rate per 25 yr generation
relativeRate(x,y)

x <- c(5600,5500)
y <- c(480,0)

# conventional growth/decline rate per 25 yr generation
100 * exp(log(y[2]/y[1])/((x[1]-x[2])/25)) - 100

# relative growth/decline rate per 25 yr generation
relativeRate(x,y)

# various random rates between 6000 and 5500 BP
x <- t(matrix(c(6000,5500),2,1000))
y <- matrix(runif(2000),1000,2)
conventional <- 100 * exp(log(y[,2]/y[,1])/((x[,1]-x[,2])/25)) - 100
relative <- relativeRate(x,y)
plot(relative, conventional)
```

`rollmean` *Rolling mean of a vector of values*

Description

Computes the rolling mean of a vector of values x. I.e., the mean of a sliding window k values wide

Usage

```
rollmean(x, k)
```

Arguments

x	A vector of values.
k	Integer width of the rolling window.

Details

Equivalent function to the rollmean function in the zoo package, but much faster.

Value

A vector of rolling means

Examples

```
x <- sample(1:1000000)
k <- 10000
rm <- rollmean(x,k)
```

`SAAD` *Radiocarbon dataset for South American Arid Diagonal (SAAD)*

Description

Dataset of terrestrial 14C dates for SAAD. Specified by using data from sites falling within the geographically contiguous 'Arid' climatic categories of the SAAD as described in the World Köppen climate classification (2006)

Usage

```
SAAD
```

Format

A data frame comprising 1527 rows and 10 columns

Source

Dataset used in Timpson et al (2020). This is a subset of the larger midholo.csv dataset compiled and published by Riris and Arroyo-Kalin (2019).

References

Timpson, A., Barberena, R., Thomas, M.G., Méndez, C., Manning, K. 2020 Directly modelling population dynamics in the South American Arid Diagonal using 14C dates. Philosophical Transactions of the Royal Society B.

Kottek, M., Grieser, J., Beck, C., Rudolf, B. & Rubel, F. 2006 World map of the Köppen-Geiger climate classification updated. Meteorologische Zeitschrift 15, 259-263.

Riris, P. & Arroyo-Kalin, M. 2019 Widespread population decline in South America correlates with mid-Holocene climate change. Scientific reports 9, 6850.

`shcal13` *Southern hemisphere 2013 calibration curve*

Description

Southern hemisphere 2013 calibration curve

Usage

```
shcal13
```

Format

A data frame comprising 5141 rows and 3 columns: cal BP, 14C BP, +/- Error.

Source

Atmospheric data from Hogg et al. (2013). Obtained from the raw shcal13.14c file downloaded from http://radiocarbon.webhost.uits.arizona.edu/node/19

References

Hogg, A.G., Hua, Q., Blackwell, P.G., Niu, M., Buck, C.E., Guilderson, T.P., Heaton, T.J., Palmer, J.G., Reimer, P.J., Reimer, R.W., 2013. SHCal13 Southern Hemisphere calibration, 0–50,000 years cal BP, Radiocarbon 55, 1889-1903.

`shcal20` *Southern hemisphere 2020 calibration curve*

Description

Southern hemisphere 2020 calibration curve

Usage

```
shcal20
```

Format

A data frame comprising 9501 rows and 3 columns: cal BP, 14C BP, +/- Error.

Source

Atmospheric data from Hogg et al. (2020). Obtained from the raw shcal20.14c file downloaded from http://intcal.org/curves/shcal20.14c

References

Hogg AG, Heaton TJ, Hua Q, Palmer JG, Turney CSM, Southon J, Bayliss A, Blackwell PG, Boswijk G, Bronk Ramsey C, Pearson C, Petchey F, Reimer P, Reimer R, Wacker L. 2020. SHCal20 Southern Hemisphere calibration, 0-55,000 years cal BP. Radiocarbon 62. doi: 10.1017/RDC.2020.59

`simulateCalendarDates` *Converts calendar dates to 14C dates*

Description

Randomly samples calendar dates from a model, including dates slightly outside the model date range to avoid edge effects

Usage

```
simulateCalendarDates(model, n)
```

Arguments

`model`	A data frame including columns 'year' and 'pdf'
`n`	The number of dates to sample.

Details

Samples n random calendar dates from a model pdf. This model must be defined in terms of a PDF vector and the corresponding calendar years. This can be provided at any preferred temporal resolution. For example, an exponential model can be provided with the PDF in annual intervals, whilst CPL model needs only the hinge points. `convertPars` will convert parameters into the required model format.

Value

Returns a vector of calendar dates

Examples

```
# under a uniform model
model <- convertPars(pars=NA, years=5000:6000,type='uniform')
sims <- simulateCalendarDates(model, 1000)
range(sims)

# simulate under an exponential model
model <- convertPars(pars=0.001, years=5000:6000,type='exp')
sims <- simulateCalendarDates(model, 1000)
range(sims)

# under a CPL model
model <- convertPars(pars=runif(5), years=5000:6000,type='CPL')
```

```
sims <- simulateCalendarDates(model, 1000)
range(sims)
```

`SPDsimulationTest` *Goodness of Fit test, using SPD simulation*

Description

Calculates the data p-value given a model

Usage

```
SPDsimulationTest(data, calcurve, calrange, pars, type, inc=5, N=20000, timeseries = NULL)
```

Arguments

`data`	A dataframe of 14C dates. Requires 'age' and 'sd', and at least one of 'site' and 'phase'. Optional 'datingType' comprising '14C' and/or anything else.
`calcurve`	A calibration curve object. Choose from intcal20 (default), shcal20, intcal13 or shcal13.
`calrange`	A vector of two calendar dates BP, giving the calendar range of CalArray. Can be in either order.
`pars`	A single vector of one parameter combination.
`type`	Choose from the following currently available `models`. Composite models can be achieved using a vector of more than one type. For example, c('norm','power') will be a composite model, where the first two parameters are the mean and SD, the 3rd and 4th parameters determine the power distribution component, for example if modelling taphonomy.
`inc`	Increments to interpolate calendar years. Default = 5
`N`	The number of simulations to generate.
`timeseries`	Only if 'type' = 'timeseries', a data frame should be provided containing columns x and y that define the timeseries as year and value respectively.

Details

The returned list provides various summary statistics and timeseries of the observed and simulated data:

timeseries: a data frame containing various CIs and:

calBP: a vector of calendar years BP.

expected.sim: a vector of the expected simulation (mean average of all N simulations).

local.sd: a vector of the local (for each year) standard deviation of all N simulations.

model: a vector of the model PDF.

SPD: a vector of the observed SPD PDF, generated from data.

index: a vector of -1,0,+1 corresponding to the SPD points that are above, within or below the 95% CI of all N simulations.

pvalue: the proportion of N simulated SPDs that have more points outside the 95% CI than the observed SPD has.

observed.stat: the summary statistic for the observed data (number of points outside the 95% CI).

simulated.stat: a vector of summary statistics (number of points outside the 95% CI), one for each simulated SPD.

n.dates.all: the total number of dates in the whole data set. Trivially, the number of rows in data.

n.dates.effective: the effective number of dates within the date range. Will be non-integer since a proportion of some dates will be outside the date range.

n.phases.all: the total number of phases in the whole data set.

n.phases.effective: the effective number of phases within the date range. Will be non-integer since a proportion of some phases will be outside the date range.

n.phases.internal: an integer subset of n.phases.all that have more than 50% of their total probability mass within the date range.

The default N = 20000 can be increased if greater precision is required, however this can be very time costly.

Value

Returns a list. See details.

Examples

```
# trivial example showing a single date can never be rejected under a uniform model:
data <- data.frame(age=6500, sd=50, phase=1, datingType='14C')
x <- SPDsimulationTest(data,
 calcurve=intcal20,
 calrange=c(6000,9000),
 pars=NULL,
 type='uniform')
print(x$pvalue)
```

`summedCalibrator` *Generates a summed probability distribution (SPD) of calibrated dates*

Description

Generates a Summed Probability Distribution (SPD). Handles both C14 and other date types e.g., thermoluminescence

Usage

```
summedCalibrator(data, CalArray, normalise = 'standard', checks = TRUE)
```

Arguments

`data`	A dataframe of 14C dates. Requires 'age' and 'sd', and at least one of 'site' and 'phase'. Optional 'datingType' to include '14C' and anything else.
`CalArray`	A 2D probability array of the calibration curve generated by `makeCalArray` containing row names and column names.
`normalise`	One of 'none', 'standard', or 'full'.
`checks`	Logical, performs various data checks if TRUE. Can be useful to set FALSE to avoid repetitive warnings if run in a loop.

Details

Uses CalArray once to simultaneously calibrate and sum all 14C dates in data. The result is equivalent to calibrating each date, then summing.

Optionally 'datingType' can be provided in the data. Only '14C' will be calibrated in the usual way, anything else is assumed to be provided in calendar time. If 'datingType' is not provided, all dates are assumed to be 14C.

If normalise is 'none', the output PD has an area equal to the total number of samples within the date range. This option is rarely required, but can be useful for example when plotting several SPDs and would also like to illustrate the relative magnitude of different datasets. However, if the date range used to generate CalArray does not encompass the entire dataset, some dates will have some probability mass outside the date range. Therefore the total probability can be non-integer and less than the sample size.

If normalise is 'standard', the output PD is normalised by the number of samples, giving an area equal to 1 provided all samples are within the date range. However, if the date range does not encompass all samples, some will have some probability mass outside the date range, resulting in the SPD area being less than 1.

If normalise is 'full', the output PD has an area equal to 1. An appropriate use includes SPD simulation testing, where it is important to ensure each simulation has the same area. In contrast, it would be absurd to apply this full normalisation to the tiny tail of a single date that is otherwise mostly outside the date range.

Value

Returns a single-column data frame of SPD probabilities. Row names are the calendar years.

Examples

```
# SPD of three 14C dates
CalArray <- makeCalArray(intcal20, calrange=c(9000,10650), inc=1 )
data <- data.frame(
 age = c(8350,8900,9350),
 sd = rep(50,3),
 datingType = rep('14C',3)
```

```
  )


# with the default normalisation the SPD area is a little under 1
# since one date is slighly outside the date range
SPD <- summedCalibrator(data, CalArray)
plotPD(SPD)
sum(SPD)

# without normalisation the total area is a little under 3
SPD <- summedCalibrator(data, CalArray, normalise='none')
plotPD(SPD)
sum(SPD)

# with full normalisation the total area is exactly 1
SPD <- summedCalibrator(data, CalArray, normalise='full')
plotPD(SPD)
sum(SPD)
```

---

summedCalibratorWrapper

Quick calibration of dates, without the need to choose a date range or generate a CalArray

---

Description

Wrapper function that easily generates and plots an SPD, at the cost of some user control

Usage

```
summedCalibratorWrapper(data, calcurve = intcal20, plot = TRUE)
```

Arguments

`data`	A dataframe of 14C dates. Requires 'age' and 'sd'.
`calcurve`	A calibration curve object. Choose from intcal20 (default), shcal20, intcal13 or shcal13.
`plot`	By default (TRUE) will plot the SPD.

Details

Function to easily plot a calibrated Summed Probability Distribution from 14C dates. Automatically chooses a sensible date range and interpolation increments. Uses these to generate CalArray internally.

Value

Returns a single-column data frame of SPD probabilities. Row names are the calendar years.

Examples

```
# SPD of two 14C dates, calibrated through intcal20 (default)
data <- data.frame(
 age=c(6562,7144),
 sd=c(44,51)
 )
x <- summedCalibratorWrapper(data)

# one date is not 14C
data <- data.frame(
 age = c(6562,7144),
 sd = c(44,51),
 datingType = c('14C','TL')
 )
x <- summedCalibratorWrapper(data)
```

---

summedPhaseCalibrator *Generates a summed probability distribution (SPD) after phasing dates*

Description

Generates a Summed Probability Distribution (SPD) after phasing dates

Usage

```
summedPhaseCalibrator(data, calcurve, calrange, inc=5, width=200)
```

Arguments

`data`	A dataframe of 14C dates. Requires ’age’ and ’sd’, and at least one of ’site’ and ’phase’. Optional ’datingType’ comprising ’14C’ and/or anything else.
`calcurve`	A calibration curve object. Choose from intcal20 (default), shcal20, intcal13 or shcal13.
`calrange`	A vector of two cal dates, giving the calendar range of CalArray. Can be in either order.
`inc`	Increments to interpolate calendar years. Default = 5.
`width`	A timespan in 14C years used to automatically bin dates if they have not been phased, i.e., ’phase’ is missing from the data. Default = 200.

Details

Wrapper function to generate an overall SPD for phased dates. Internally this first generates an SPD for each phase. Data may be phased already, alternatively if ’phase’ is not provided, this function will automatically bin dates into phases, see `phaseCalibrator`. Each phase’s distribution is then summed, and the final SPD is normalised to unity.

Optionally 'datingType' can be provided in the data. Only '14C' will be calibrated in the usual way, anything else is assumed to be provided in calendar time. If 'datingType' is not provided, all dates are assumed to be 14C.

Value

Returns a single-column data frame of SPD probabilities. Row names are the calendar years.

Examples

```
data <- subset(SAAD, site %in% c('Carrizal','Pacopampa'))
SPD <- summedPhaseCalibrator(data, shcal20, c(2000,6000))
plotPD(SPD)
```

---

toy *Toy population model*

Description

Data frame of a toy population model. Provides a discretised PDF across the time range of 7.5kyr to 5.5kyr BP. The model PDF outside this range is zero. Comprises two columns: year, pdf

Usage

```
toy
```

Format

A data frame comprising 4 rows and 2 columns

Source

Toy used in Timpson et al (2020).

References

Timpson, A., Barberena, R., Thomas, M.G., Méndez, C., Manning, K. 2020 Directly modelling population dynamics in the South American Arid Diagonal using 14C dates. Philosophical Transactions of the Royal Society B.

uncalibrateCalendarDates

Converts calendar dates to 14C dates

Description

Randomly samples a 14C date from the calibration curve error ribbon, at the corresponding calendar date

Usage

```
uncalibrateCalendarDates(dates, calcurve)
```

Arguments

`dates`	A vector of calendar dates.
`calcurve`	A calibration curve object. Choose from intcal20 (default), shcal20, intcal13 or shcal13.

Details

Conceptually this can be thought of as the reverse process of calibrating a 14C date into calendar time, however 'uncalibrating' is a misnomer as the full calibrated PD is not used. Instead, it uses a vector of calendar point estimates, and randomly samples 14C dates from the calibration curve error ribbon, at the corresponding calendar dates. Therefore values will differ each time.

Value

Returns a vector of 14C dates

Examples

```
uncalibrateCalendarDates(c(4500,5000), shcal20)

# note the date outside the calcurve range has a 1 to 1 mapping between cal and c14 time
uncalibrateCalendarDates(c(4500,70000), intcal20)

# however, a soft fade is performed between the end of the calcurve and 60000
uncalibrateCalendarDates(c(4500,58000), intcal20)
```

Index