Package 'ShellChron'

July 5, 2021

Title Builds Chronologies from Oxygen Isotope Profiles in Shells

Version 0.4.0

Description Takes as input a stable oxygen isotope (d18O) profile measured in growth direction (D) through a shell + uncertainties in both variables (d18O_err & D_err). It then models the seasonality in the d18O record by fitting a combination of a growth and temperature sine wave to year-length chunks of the data (see Judd et al., (2018) <doi:10.1016/j.palaeo.2017.09.034>). This modeling is carried out along a sliding window through the data and yields estimates of the day of the year (Julian Day) and local growth rate for each data point. Uncertainties in both modeling routine and the data itself are propagated and pooled to obtain a confidence envelope around the age of each data point in the shell. The end result is a shell chronology consisting of estimated ages of shell formation relative to the annual cycle with their uncertainties. All formulae in the package serve this purpose, but the user can customize the model (e.g. number of days in a year and the mineralogy of the shell carbonate) through input parameters.

Imports rtop (>= 0.5.14), zoo (>= 1.8.7), ggplot2 (>= 3.2.1), ggpubr (>= 0.4.0), tidyr (>= 1.1.1), scales (>= 1.1.0), dplyr, magrittr

License GPL-3

URL https://github.com/nielsjdewinter/ShellChron

BugReports https://github.com/nielsjdewinter/ShellChron/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown

NeedsCompilation no

Author Niels de Winter [aut, cre] (<https://orcid.org/0000-0002-1453-5407>)

Maintainer Niels de Winter <niels_de_winter@live.nl>

Repository CRAN

Date/Publication 2021-07-05 12:40:02 UTC

R topics documented:

`age_corr` *Function that corrects chronologies for sudden jumps in time*

Description

Some occurrences in the model results can lead the CumDY function to detect extra year transitions, resulting in sudden jumps in the shell chronology or a start of the chronology at an age beyond 1 year. This function removes these sharp transitions and late onset by adding or subtracting whole years to the age result.

Usage

```
age_corr(resultarray, T_per = 365, plot = TRUE, agecorrection = TRUE)
```

Arguments

`resultarray`	Array containing the full results of the optimized growth model
`T_per`	The period length of one year (in days)
`plot`	Should the results be plotted? (/codeTRUE/FALSE)
`agecorrection`	Correct for jumps in age (/codeTRUE) or only for starting time (/codeFALSE)

Value

An updated and corrected version of `resultarray`

References

package dependencies: ggplot2 3.2.1

Examples

```
testarray <- array(NA, dim = c(20, 16, 9)) # Create empty array
# with correct third dimension
windowfill <- seq(10, 100, 10) # Create dummy simulation data
# (ages) to copy through the array
for(i in 6:length(testarray[1, , 1])){
    testarray[, i, 3] <- c(windowfill,
        rep(NA, length(testarray[, 1, 3]) - length(windowfill)))
    windowfill <- c(NA, windowfill + 31)
}
testarray2 <- age_corr(testarray, 365, FALSE, FALSE) # Apply function on
array
```

`cumdy` *Function to detect year transitions and calculate cumulative age of model results*

Description

Takes the result of iterative growth modeling and transforms data from Julian Day (0 - 365) to cumulative day of the shell age by detecting where transitions from one year to the next occur and adding full years (365 days) to simulations in later years.

Usage

```
cumdy(resultarray, threshold = 5, plotyearmarkers = TRUE)
```

Arguments

`resultarray`	Array containing the full results of the optimized growth model
`threshold`	Artificial threshold value used to recognize peaks in occurrences of year transitions (default = 5)
`plotyearmarkers`	Should the location of identified year transitions be plotted? `TRUE/FALSE`

Value

A new version of the resultarray with Julian Day model estimates replaced by estimates of cumulative age of the record in days.

References

package dependencies: zoo 1.8.7

Examples

```
testarray <- array(NA, dim = c(20, 16, 9)) # Create empty array
# with correct third dimension
windowfill <- seq(50, 500, 50) # Create dummy simulation data
# (ages) to copy through the array
for(i in 6:length(testarray[1, , 1])){
    testarray[, i, 3] <- c(windowfill, rep(NA, length(testarray[, 1, 3]) -
        length(windowfill)))
    windowfill <- c(NA, (windowfill + 51) %% 365)
}
testarray[, 1, 3] <- seq(1, length(testarray[, 1, 3]), 1) # Add
# dummy /code{D} column.
testarray2 <- cumdy(testarray, 3, FALSE) # Apply function on array
```

---

cumulative_day — *Function to detect year transitions and calculate cumulative age of model results*

Description

Takes the result of iterative growth modeling and transforms data from Julian Day (0 - 365) to cumulative day of the shell age by detecting where transitions from one year to the next occur and adding full years (365 days) to simulations in later years.

Usage

```
cumulative_day(
  resultarray,
  plotyearmarkers = TRUE,
  export_peakid = TRUE,
  path = tempdir()
)
```

Arguments

`resultarray`	Array containing the full results of the optimized growth model
`plotyearmarkers`	Should the location of identified year transitions be plotted? `TRUE/FALSE`
`export_peakid`	Should the result of peak identification be plotted? `TRUE/FALSE`
`path`	Export path (defaults to tempdir())

Value

A new version of the Julian Day tab of the resultarray with Julian Day model estimates replaced by estimates of cumulative age of the record in days.

References

package dependencies: zoo 1.8.7; scales 1.1.0; graphics function dependencies: peakid

Examples

```
testarray <- array(NA, dim = c(40, 36, 9)) # Create empty array
# with correct third dimension
windowfill <- seq(50, 500, 50) %% 365 # Create dummy simulation data
# (ages) to copy through the array
for(i in 6:length(testarray[1, , 1])){
    testarray[, i, 3] <- c(windowfill, rep(NA, length(testarray[, 1, 3]) -
        length(windowfill)))
    windowfill <- c(NA, (windowfill + 51) %% 365)
}
# Add dummy /code{D} column.
testarray[, 1, 3] <- seq(1, length(testarray[, 1, 3]), 1)
# Add dummy YEARMARKER column
testarray[, 3, 3] <- c(0, rep(c(0, 0, 0, 0, 0, 0, 1), 5), 0, 0, 0, 0)
# Add dummy d18Oc column
testarray[, 2, 3] <- sin((2 * pi * (testarray[, 1, 3] - 8 + 7 / 4)) / 7)
testarray2 <- suppressWarnings(cumulative_day(testarray, FALSE, FALSE, tempdir()))
# Apply function on array
```

`d18O_model` *Function to convert SST data to d18O*

Description

Takes a matrix of SST data (in degrees C) against time (in days), information about the d18O value (in permille VSMOW) of the water and how it changes through the year and the transfer function used for of the record (e.g. Kim and O'Neil, 1997 or Grossman and Ku, 1986). Converts the SST data to d18O data using the supplied empirical transfer function.

Usage

```
d18O_model(SST, d18Ow = 0, transfer_function = "KimONeil97")
```

Arguments

`SST`	Matrix with a time column (values in days) and an SST column (values in degrees C)

`d18Ow` Either a single value (constant d18Ow) or a vector of length equal to the period in SST data (365 days by default) containing information about seasonality in d18Ow. Defaults to constant d18Ow of 0 permille VSMOW (the modern mean ocean value)

`transfer_function`
String containing the name of the transfer function (for example: `"KimONeil97"` or `"GrossmanKu86"`). Defaults to Kim and O'Neil (1997).

Value

A vector containing d18O values for each SST value in `"SST"`

References

Grossman, E.L., Ku, T., Oxygen and carbon isotope fractionation in biogenic aragonite: temperature effects, *Chemical Geology* **1986**, *59.1*, 59-74. doi: 10.1016/01689622(86)900576 Kim, S., O'Niel, J.R., Equilibrium and nonequilibrium oxygen isotope effects in synthetic carbonates, *Geochimica et Cosmochimica Acta* **1997**, *61.16*, 3461-3475. doi: 10.1016/S00167037(97)001695 Dettman, D.L., Reische, A.K., Lohmann, K.C., Controls on the stable isotope composition of seasonal growth bands in aragonitic fresh-water bivalves (Unionidae), *Geochimica et Cosmochimica Acta* **1999**, *63.7-8*, 1049-1057. doi: 10.1016/S00167037(99)000204 Brand, W.A., Coplen, T.B., Vogl, J., Rosner, M., Prohaska, T., Assessment of international reference materials for isotope-ratio analysis (IUPAC Technical Report), *Pure and Applied Chemistry* **2014**, *86.3*, 425-467. doi: 10.1515/pac-20131023

Examples

```
# Create dummy SST data
t <- seq(1, 40, 1)
T <- sin((2 * pi * (seq(1, 40, 1) - 8 + 10 / 4)) / 10)
SST <- cbind(t, T)
# Run d18O model function
d18O <- d18O_model(SST, 0, "KimONeil97")
```

`data_import` *Function to import d18O data and process yearmarkers and calculation windows*

Description

Takes the name of a file that is formatted according to the standard format and converts it to an object to be used later in the model. In doing so, the function also reads the user-provided yearmarkers in the file and uses them as a basis for the length of windows used throughout the model. This ensures that windows are not too short and by default contain at least one year of growth for modeling.

Usage

```
data_import(file_name)
```

Arguments

`file_name` Name of the file that contains sampling distance and d18O data. Note that sampling distance should be given in micrometers, because the SCEUA model underperforms when the growth rate figures are very small (<0.1 mm/day).

Value

A list containing an object with the original data and details on the position and length of modeling windows

Examples

```
importlist <- data_import(file_name = system.file("extdata",
    "Virtual_shell.csv", package = "ShellChron")) # Run function on attached
    # dummy data

# Bad data file lacking YEARMARKER column
## Not run: importlist <- data_import(file_name = system.file("extdata",
    "Bad_data.csv", package = "ShellChron"))
## End(Not run)
```

---

export_results *Function to merge and export the results of the ShellChron model*

Description

Takes the input data and model results and reformats them to tables of key parameters such as growth rate and shell age for each datapoint for easy plotting. This final function also combines uncertainties in the model result arising from uncertainties in input data (provided by the user) and uncertainties of the model (from overlapping modeling windows). Includes some optional plotting options.

Usage

```
export_results(
  path = getwd(),
  dat,
  resultarray,
  parmat,
  MC = 1000,
  dynwindow,
  plot = FALSE,
  plot_export = TRUE,
  export_raw = FALSE
)
```

Arguments

`path`	Path where result files are exported
`dat`	Matrix containing the input data
`resultarray`	Array containing the full results of the optimized growth model
`parmat`	Matrix listing all optimized growth rate and SST parameters used to model d18O in each data window
`MC`	Number of Monte Carlo simulations to apply for error propagation. Default = 1000
`dynwindow`	Information on the position and length of modeling windows
`plot`	Should an overview of the results of modeling be plotted? `TRUE/FALSE`
`plot_export`	Should the overview plot be exported as a PDF file? `TRUE/FALSE`
`export_raw`	Export tables containing all raw model results before being merged into tidy tables? `TRUE/FALSE`

Value

CSV tables of model results in the current working directory + optional plots in PDF format

References

package dependencies: tidyverse 1.3.0; ggpubr 0.4.0; magrittr function dependencies: sd_wt

Examples

```
# Create dummy input data column by column
dat <- as.data.frame(seq(1000, 40000, 1000))
colnames(dat) <- "D"
dat$d18Oc <- sin((2 * pi * (seq(1, 40, 1) - 8 + 7 / 4)) / 7)
dat$YEARMARKER <- c(0, rep(c(0, 0, 0, 0, 0, 0, 1), 5), 0, 0, 0, 0)
dat$D_err <- rep(100, 40)
dat$d18Oc_err <- rep(0.1, 40)

testarray <- array(NA, dim = c(40, 36, 9)) # Create empty array
# with correct third dimension
windowfill <- seq(50, 500, 50) %% 365 # Create dummy simulation data
# (ages) to copy through the array
for(i in 6:length(testarray[1, , 1])){
    testarray[, i, 3] <- c(windowfill, rep(NA, length(testarray[, 1, 3]) -
        length(windowfill)))
    windowfill <- c(NA, (windowfill + 51) %% 365)
}
# Add dummy /code{D} column.
testarray[, 1, 3] <- seq(1, length(testarray[, 1, 3]), 1)
# Add dummy YEARMARKER column
testarray[, 3, 3] <- c(0, rep(c(0, 0, 0, 0, 0, 0, 1), 5), 0, 0, 0, 0)
# Add dummy d18Oc column
testarray[, 2, 3] <- sin((2 * pi * (testarray[, 1, 3] - 8 + 7 / 4)) / 7)
# Create dummy seasonality data
```

```
seas <- as.data.frame(seq(1, 365, 1))
colnames(seas) <- "t"
seas$SST <- 15 + 10 * sin((2 * pi * (seq(1, 365, 1) - 182.5 +
    365 / 4)) / 365)
seas$GR <- 10 + 10 * sin((2 * pi * (seq(1, 365, 1) - 100 + 365 / 4)) / 365)
seas$d18O <- (exp((18.03 * 1000 / (seas$SST + 273.15) - 32.42) / 1000) - 1) *
    1000 + (0.97002 * 0 - 29.98)
# Apply dummy seasonality data to generate other tabs of testarray
testarray[, , 1] <- seas$d18O[match(testarray[, , 3], seas$t)] # d18O values
tab <- testarray[, , 1]
tab[which(!is.na(tab))] <- 0.1
testarray[, , 2] <- tab # dummy d18O residuals
testarray[, , 4] <- seas$GR[match(testarray[, , 3], seas$t)] # growth rates
testarray[, , 5] <- seas$SST[match(testarray[, , 3], seas$t)] # temperature
tab[which(!is.na(tab))] <- 0.1
testarray[, , 6] <- tab # dummy d18O SD
tab[which(!is.na(tab))] <- 20
testarray[, , 7] <- tab # dummy time SD
tab[which(!is.na(tab))] <- 3
testarray[, , 8] <- tab # dummy GR SD
tab[which(!is.na(tab))] <- 1
testarray[, , 9] <- tab # dummy temperature SD
darray <- array(rep(as.matrix(dat), 9), dim = c(40, 5, 9))
testarray[, 1:5, ] <- darray

# Create dummy dynwindow data
dynwindow <- as.data.frame(seq(1, 31, 1))
colnames(dynwindow) <- "x"
dynwindow$y <- rep(10, 31)

dimnames(testarray) <- list(
    paste("sample", 1:length(testarray[, 1, 3])),
    c(colnames(dat), paste("window", 1:length(dynwindow$x))),
    c("Modeled_d18O",
        "d18O_residuals",
        "Time_of_year",
        "Instantaneous_growth_rate",
        "Modeled temperature",
        "Modeled_d18O_SD",
        "Time_of_Year_SD",
        "Instantaneous_growth_rate_SD",
        "Modeled_temperature_SD")
)

# Set parameters
G_amp <- 20
G_per <- 365
G_pha <- 100
G_av <- 15
G_skw <- 70
T_amp <- 20
T_per <- 365
T_pha <- 150
```

```
T_av <- 15
pars <- c(T_amp, T_pha, T_av, G_amp, G_pha, G_av, G_skw)
parsSD <- c(3, 10, 3, 5, 10, 3, 5) # Artificial variability in parameters
parmat <- matrix(rnorm(length(pars) * length(dynwindow$x)), nrow =
    length(pars)) * parsSD + matrix(rep(pars, length(dynwindow$x)),
    nrow = length(pars))
rownames(parmat) <- c("T_amp", "T_pha", "T_av", "G_amp", "G_pha", "G_av",
    "G_skw")
# Run export function
test <- export_results(path = tempdir(),
    dat,
    testarray,
    parmat,
    MC = 1000,
    dynwindow,
    plot = FALSE,
    plot_export = FALSE,
    export_raw = FALSE)
```

`growth_model` *Function that models a d18O curve through SST and GR sinusoids*

Description

The core function of the ShellChron growth model. Uses growth rate and SST (Sea Surface Temperature) sinusoids to model d18O data to be matched with the input. In the ShellChron modeling routine, this function is optimized using the SCEUA algorithm and applied on sliding windows through the dataset to estimate the age of each datapoint

Usage

```
growth_model(
  pars,
  T_per = 365,
  G_per = 365,
  years = 1,
  t_int = 1,
  transfer_function = "KimONeil97",
  d18Ow = "default",
  Dsam,
  Osam,
  t_maxtemp = 182.5,
  plot = FALSE,
  MC = 1000,
  D_err = NULL,
  O_err = NULL,
  return = "SSR"
)
```

Arguments

- `pars` — List of parameters for temperature and growth rate sinusoids `pars <-c(T_amp,T_pha,T_av,G_amp,G_pha`
- `T_per` — Period of SST sinusoid (in days; default = 365)
- `G_per` — Period of growth rate sinusoid (in days; default = 365)
- `years` — Number of years to be modeled (default = 1)
- `t_int` — Time interval (in days; default = 1)
- `transfer_function` — Transfer function used to convert d18Oc to temperature data.
- `d18Ow` — Either a single value (constant d18Ow) or a vector of length equal to the period in SST data (365 days by default) containing information about seasonality in d18Ow. Defaults to constant d18Ow of 0 permille VSMOW (the modern mean ocean value)
- `Dsam` — Vector of `D` values serving as input (keep unit consistent throughout model)
- `Osam` — Vector of `d18Oc` values serving as input (in permille VPDB)
- `t_maxtemp` — Timing of the warmest day of the year (in julian day; default = 182.5, or May 26th halfway through the year)
- `plot` — Should results of modeling be plotted? `TRUE/FALSE`
- `MC` — Number of Monte Carlo simulations to apply for error propagation Default = 1000
- `D_err` — OPTIONAL: Vector containing errors on `Dsam`
- `O_err` — OPTIONAL: Vector containing errors on `Osam`
- `return` — String indicating whether to return just the Sum of Squared Residuals ("SSR") or a matrix containing the results of the model and the propagated uncertainties (if applicable)

Value

Depending on the value of the "return" parameter either a single value representing the Sum of Squared Residuals ("SSR") as a measure for the closeness of the match between modeled d18O and input values, or a matrix containing the full result of the modeling including propagated uncertainties if applicable.

References

package dependencies: ggplot2 3.2.1 function dependencies: temperature_curve, d18O_model, growth_rate_curve, mc_err_orth

doi: 10.1016/j.palaeo.2017.09.034

Examples

```
# Set parameters
G_amp <- 20
G_per <- 365
G_pha <- 100
```

```
G_av <- 15
G_skw <- 70
T_amp <- 20
T_per <- 365
T_pha <- 150
T_av <- 15
pars <- c(T_amp, T_pha, T_av, G_amp, G_pha, G_av, G_skw)
d18Ow <- 0
# Create dummy data
Dsam <- seq(1, 40, 1)
Osam <- sin((2 * pi * (seq(1, 40, 1) - 8 + 30 / 4)) / 30)
# Test returning residual sum of squares for optimization
SSR <- growth_model(pars, T_per, G_per, Dsam = Dsam, Osam = Osam,
    return = "SSR")
# Test returning full model result
resmat <- growth_model(pars, T_per, G_per, Dsam = Dsam, Osam = Osam,
    return = "result")
```

`growth_rate_curve` *Function that creates a skewed sinusoidal growth rate (GR) curve from a list of parameters*

Description

Takes the specified parameters for amplitude, period, phase, average value and skewness factor as well as the number of years specified and the time interval. It then creates a skewed sinusoid based on the boundary conditions. The skewness factor (`G_skw`) determines whether the sinusoid is skewed towards the front (`G_skw < 50`) or the back of the annual peak in growth rate (`G_skw > 50`). Used as intermediate step during iterative modeling.

Usage

```
growth_rate_curve(G_par, years = 1, t_int = 1)
```

Arguments

`G_par`	List of four parameters describing (in order) amplitude (`G_amp`; in micrometer/day), period (`G_per`; in days), phase (`G_pha` in day of the year), average growth rate (`G_av`; in micrometer/day) and the skewness factor (`G_skw` between 0 and 100)
`years`	Length of the preferred sinusoid in number of years (defaults to 1)
`t_int`	Time interval of sinusoidal record (in days)

Value

A matrix containing columns for time (in days) and GR (in micrometer/day)

References

doi: 10.1016/j.palaeo.2017.09.034

Examples

```
# Set parameters
G_amp <- 20
G_per <- 365
G_pha <- 100
G_av <- 15
G_skw <- 70
G_par <- c(G_amp, G_per, G_pha, G_av, G_skw)
# Run GR model function
GR <- growth_rate_curve(G_par, 1, 1)
```

---

mc_err_form — *Function that propagates measurement uncertainty through model results*

Description

Function to propagate combined errors on `x` (= `Dsam`) and `y` (= `Osam`) on the modeled X (= `D`) and Y (= `d18Oc`) values by means of projection of uncertainties through the modeled `X-Y` relationship

Usage

```
mc_err_form(x, x_err, y, y_err, X, Y, MC = 1000)
```

Arguments

`x`	Vector of `x` values of input data
`x_err`	Vector of uncertainties on `x` values
`y`	Vector of `y` values of input data
`y_err`	Vector of uncertainties on `y` values
`X`	Vector of modeled `X` values on which the uncertainty is to be projected
`Y`	Matrix of modeled x and `Y` values
`MC`	Number of Monte Carlo simulations to apply for error propagation Default = 1000

Details

Note: projection leads to large uncertainties on shallow parts of the `XY` curve

Value

A vector listing the standard deviations of propagated errors propagated on all `X` values.

Examples

```
# Create dummy data for input
x <-  seq(1, 40, 1)
x_err <- rep(0.1, 40)
y <- sin((2 * pi * (seq(1, 40, 1) - 8 + 30 / 4)) / 30)
y_err <- rep(0.1, 40)
X <- seq(1.5, 39.5, 1)
Y <- cbind(seq(1, 39, 1), 0.9 * sin((2 * pi * (seq(1, 39, 1) - 9 +
    25 / 4)) / 25))
# Run function
result <- mc_err_form(x, x_err, y, y_err, X, Y, 1000)
```

---

mc_err_orth — *Function that propagates measurement uncertainty through model results*

Description

Function to propagate combined errors on `x` (= `Dsam`) and `y` (= `Osam`) on the modeled X (= `D`) and Y (= `d18Oc`) values by means of orthogonal projection of uncertainty on `x` and `y` onto the model curve

Usage

```
mc_err_orth(x, x_err, y, y_err, X, Y, MC = 1000)
```

Arguments

`x`	Vector of `x` values of input data
`x_err`	Vector of uncertainties on `x` values
`y`	Vector of `y` values of input data
`y_err`	Vector of uncertainties on `y` values
`X`	Vector of modeled `X` values on which the uncertainty is to be projected
`Y`	Matrix of modeled `X` and `Y` values
`MC`	Number of Monte Carlo simulations to apply for error propagation Default = 1000

Value

A vector listing the standard deviations of propagated errors propagated on all `X` values.

Examples

```
# Create dummy data for input
x <-  seq(1, 40, 1)
x_err <- rep(0.1, 40)
y <- sin((2 * pi * (seq(1, 40, 1) - 8 + 30 / 4)) / 30)
y_err <- rep(0.1, 40)
X <- seq(1.5, 39.5, 1)
Y <- cbind(seq(1, 39, 1), 0.9 * sin((2 * pi * (seq(1, 39, 1) - 9 +
    25 / 4)) / 25))
# Run function
result <- mc_err_orth(x, x_err, y, y_err, X, Y, 1000)
```

---

mc_err_proj — *Function that propagates measurement uncertainty through model results*

Description

Function to propagate combined errors on x (= Dsam) and y (= Osam) on the modeled X (= D) and Y (= d18Oc) values by means of direct projection of y–uncertainty on x and then combine the errors on both in the x domain

Usage

```
mc_err_proj(x, x_err, y, y_err, X, Y, MC = 1000)
```

Arguments

x	Vector of x values of input data
x_err	Vector of uncertainties on x values
y	Vector of y values of input data
y_err	Vector of uncertainties on y values
X	Vector of modeled X values on which the uncertainty is to be projected
Y	Matrix of modeled x and Y values
MC	Number of Monte Carlo simulations to apply for error propagation Default = 1000

Details

Note: projection y_err on x_err leads to large X errors on shallow slopes due to numerical calculation of fist derivative.

Value

A vector listing the standard deviations of propagated errors propagated on all X values.

Examples

```
# Create dummy data for input
x <-  seq(1, 40, 1)
x_err <- rep(0.1, 40)
y <- sin((2 * pi * (seq(1, 40, 1) - 8 + 30 / 4)) / 30)
y_err <- rep(0.1, 40)
X <- seq(1.5, 39.5, 1)
Y <- cbind(seq(1, 39, 1), 0.9 * sin((2 * pi * (seq(1, 39, 1) - 9 +
    25 / 4)) / 25))
# Run function
result <- mc_err_proj(x, x_err, y, y_err, X, Y, 1000)
```

`peakid` *Function that identifies peaks in a dataset*

Description

Developed by William A. Huber

Usage

```
peakid(x, y, w = 1, ...)
```

Arguments

`x`	Vector of `x` values of input data
`y`	Vector of `y` values of input data
`w`	Window size for smoothing data
`...`	Additional arguments to be passed into LOESS function

Value

A vector listing the standard deviations of propagated errors propagated on all `X` values.

References

package dependencies: zoo 1.8.7

Huber, W.A., Data Smoothing and Peak Detection, Rpubs, Last accessed: December 8th, 2020. https://rpubs.com/mengxu/peak_detection

See Also

https://rpubs.com/mengxu/peak_detection

Examples

```
# Create dummy periodic data
x <- seq(1, 100, 1)
y <- sin((2 * pi * (seq(1, 100, 1) - 8 + 20 / 4)) / 20)
# Run peakid function
result <- peakid(x, y, w = 20)
```

---

run_model   *Function that optimizes sinusoid parameters to fit d18O data*

Description

The second core function of the ShellChron growth model. Loops through all data windows and uses the `growth_model` function to create d18O series that match the input data. This step is iterated and optimized (minimizing the Sum of Squared Residuals) through the SCEUA algorithm (by Duan et al., 1992) which finds the optimal input parameters to the growth rate and Sea Surface Temperature (SST) sinusoids to simulate d18O data.

Usage

```
run_model(
  dat,
  dynwindow,
  transfer_function = "KimONeil97",
  d18Ow = 0,
  T_per = 365,
  G_per = 365,
  t_int = 1,
  t_maxtemp = 182.5,
  SCEUApar = c(1, 25, 10000, 5, 0.01, 0.01),
  sinfit = TRUE,
  MC = 1000,
  plot = FALSE
)
```

Arguments

`dat` Matrix containing the input data

`dynwindow` Information on the position and length of modeling windows

`transfer_function`
Transfer function used to convert d18Oc to temperature data.

`d18Ow` Either a single value (constant d18Ow) or a vector of length equal to the period in SST data (365 days by default) containing information about seasonality in d18Ow. Defaults to constant d18Ow of 0 permille VSMOW (the modern mean ocean value)

`T_per` Period of SST sinusoid (in days; default = 365)

G_per	Period of growth rate sinusoid (in days; default = 365)
t_int	Time interval (in days; default = 1)
t_maxtemp	Timing of the warmest day of the year (in julian day; default = 182.5, or May 26th halfway through the year)
SCEUApar	Parameters for SCEUA optimization (iniflg, ngs, maxn, kstop pcento, peps). For details, refer to Duan et al. (1992) in references
sinfit	Apply sinusoidal fitting to guess initial parameters for SCEUA optimization? TRUE/FALSE
MC	Number of Monte Carlo simulations to apply for error propagation Default = 1000
plot	Should results of modeling be plotted? TRUE/FALSE

Value

A list containing the resultarray which contains the full result of all simulations on each data window and the parmat listing all optimized growth rate and SST parameters used to model d18O in each data window

References

package dependencies: ggplot2 3.2.1; rtop 0.5.14 Function dependencies: sinreg, d18O_model, growth_model

doi: 10.1029/91WR02985

See Also

Duan, Qingyun, Soroosh Sorooshian, and Vijai Gupta. "Effective and efficient global optimization for conceptual rainfall runoff models." Water resources research 28.4 (1992): 1015-1031. https://doi.org/10.1029/91WR02985

Examples

```
# Create dummy input data column by column
dat <- as.data.frame(seq(1000, 40000, 1000))
colnames(dat) <- "D"
dat$d18Oc <- sin((2 * pi * (seq(1, 40, 1) - 8 + 7 / 4)) / 7)
dat$YEARMARKER <- c(0, rep(c(0, 0, 0, 0, 0, 0, 1), 5), 0, 0, 0, 0)
dat$D_err <- rep(100, 40)
dat$d18Oc_err <- rep(0.1, 40)
# Create dummy dynwindow data
dynwindow <- as.data.frame(seq(1, 29, 2))
colnames(dynwindow) <- "x"
dynwindow$y <- rep(12, 15)
# Run model function
resultlist <- run_model(dat = dat,
    dynwindow = dynwindow,
    transfer_function = "KimONeil97",
    d18Ow = 0,
    T_per = 365,
```

```
    G_per = 365,
    t_int = 1,
    t_maxtemp = 182.5,
    SCEUApar = c(1, 25, 10000, 5, 0.01, 0.01),
    sinfit = TRUE,
    MC = 1000,
    plot = FALSE)
```

---

sd_wt *Function to calculate weighted standard deviation*

Description

Calculates the standard deviation of a weighted sample set while propagating sample weights through the calculation.

Usage

```
sd_wt(x, w, na.rm = FALSE)
```

Arguments

x	Vector containing the values in the set
w	Vector containing the weights to each value (in the same order as x the optimized growth model
na.rm	Should NA values be removed from the set prior to calculation? TRUE/FALSE

Value

The standard deviation of the weighted set of x values

Examples

```
# Create dummy data
x <- seq(1, 10, 0.5)
w <- c(seq(0.1, 1, 0.1), seq(0.9, 0.1, -0.1))
SDw <- sd_wt(x, w, na.rm = TRUE) # Run the function
```

sinreg — *Function that carries out a sinusoidal regression*

Description

Fits a sinusoid through data provided as an x and y vector and returns a list containing both the fitted curve and the parameters of that curve. Used to produce initial values for modeling data windows and later to find peaks in modeled julian day values to align the result to a cumulative age timeline.

Usage

```
sinreg(x, y, fixed_period = NA, plot = FALSE)
```

Arguments

x	Vector of x values of input data
y	Vector of y values of input data
fixed_period	Optional variable for fixing the period of the sinusoid in the depth domain. Defaults to NA, period is not fixed. Supply a single value to fix the period.
plot	Should the fitting result be plotted? TRUE/FALSE

Value

A list containing a vector of parameters of the fitted sinusoid and the fitted values belonging to each x value. Fitting parameters: I = the mean annual value of the sinusoid (height) A = the amplitude of the sinusoid Dper = the period of the sinusoid in x domain peak = the location of the peak in the sinusoid R2adj = the adjusted R^2 value of the fit p = the p-value of the fit

Examples

```
# Create dummy data
x <- seq(1000, 11000, 1000)
y <- sin((2 * pi * (seq(1, 11, 1) - 8 + 7 / 4)) / 7)
sinlist <- sinreg(x, y, plot = FALSE) # Run the function
```

temperature_curve — *Function that creates a sinusoidal Sea Surface Temperature (SST) curve from a list of parameters*

Description

Takes the specified parameters for amplitude, period, phase and average value as well as the number of years specified and the time interval. It then creates a sinusoid based on the boundary conditions. Used as intermediate step during iterative modeling.

Usage

```
temperature_curve(T_par, years = 1, t_int = 1)
```

Arguments

T_par	List of four parameters describing (in order) amplitude (T_amp; in degrees C), period (T_per; in days), phase (T_pha in day of the year) and average temperature (T_av; in degrees C)
years	Length of the preferred sinusoid in number of years (defaults to 1)
t_int	Time interval of sinusoidal record (in days)

Value

A matrix containing columns for time (in days) and SST (in degrees C)

Examples

```
# Set parameters
T_amp <- 20
T_per <- 365
T_pha <- 150
T_av <- 15
T_par <- c(T_amp, T_per, T_pha, T_av)
SST <- temperature_curve(T_par, 1, 1) # Run the function
```

Virtual_shell *Virtual input data for ShellChron*

Description

A dataset containing data used to test the ShellChron functions. Generated using the code in "Generate_Virtual_shell.r" in data-raw

Usage

```
Virtual_shell
```

Format

A data frame with 80 rows and 5 variables:

D Depth, in μm along the virtual record

d18Oc stable oxygen isotope value, in permille VPDB

D_err Depth uncertainty, in μm

d18Oc_err stable oxygen isotope value uncertainty, in permille VPDB

YEARMARKER "1" marking year transitions ...

Source

See code to generate data in data-raw Modified after virtual data described in de Winter et al., 2021 https://doi.org/gk98

---

wrap_function *Full ShellChron workflow wrapped in a single function*

Description

Takes starting parameters and names of input files and directory and runs through all the steps of the ShellChron model. Function includes options for plotting and exporting raw data, which are parsed into underlying formulae.

Usage

```
wrap_function(
  path = getwd(),
  file_name,
  transfer_function = "KimONeil97",
  t_int = 1,
  T_per = 365,
  d18Ow = 0,
  t_maxtemp = 182.5,
  SCEUApar = c(1, 25, 10000, 5, 0.01, 0.01),
  sinfit = TRUE,
  MC = 1000,
  plot = TRUE,
  plot_export = TRUE,
  export_raw = FALSE,
  export_path = getwd()
)
```

Arguments

path	String containing the path to the directory that contains the input data.
file_name	Name of the file that contains d18O data
transfer_function	String containing the name of the transfer function. Defaults to Kim and O'Neil, 1997.
t_int	Time interval (in days; default = 1)
T_per	Period of SST sinusoid (in days; default = 365)
d18Ow	Either a single value (constant d18Ow) or a vector of length equal to the period in SST data (365 days by default) containing information about seasonality in d18Ow. Defaults to constant d18Ow of 0 permille VSMOW (the modern mean ocean value)

t_maxtemp	Timing of the warmest day of the year (in julian day; default = 182.5, or May 26th halfway through the year)
SCEUApar	Parameters for SCEUA optimization (iniflg, ngs, maxn, kstop pcento, peps) For details, refer to Duan et al. (1992) in references
sinfit	Apply sinusoidal fitting to guess initial parameters for SCEUA optimization? TRUE/FALSE
MC	Number of Monte Carlo simulations to apply for error propagation. Default = 1000
plot	Should an overview of the results of modeling be plotted? TRUE/FALSE
plot_export	Should the overview plot be exported as a PDF file? TRUE/FALSE
export_raw	Export tables containing all raw model results before being merged into tidy tables? TRUE/FALSE
export_path	Path where result files are exported

Value

CSV tables of model results in the current working directory, optional plots in PDF format and list object of model results for further processing in the R workspace.

References

function dependencies: data_import, run_model, cumulative_day, export_results

Examples

```
# find attached dummy data
example <- wrap_function(path = getwd(),
    file_name = system.file("extdata", "Virtual_shell.csv",
    package = "ShellChron"),
    transfer_function = "KimONeil97",
    t_int = 1,
    T_per = 365,
    d18Ow = 0,
    t_maxtemp = 182.5,
    SCEUApar = c(1, 25, 10000, 5, 0.01, 0.01),
    sinfit = TRUE,
    MC = 1000,
    plot = FALSE,
    plot_export = FALSE,
    export_raw = FALSE,
    export_path = tempdir()) # Run function
```

Index