Package 'dplR'

January 31, 2025

Encoding UTF-8

Type Package

Title Dendrochronology Program Library in R

Version 1.7.8

Copyright Authors and file inst/COPYRIGHTS

Depends R (>= 3.5.0)

Imports graphics, grDevices, grid, stats, utils, lattice (>= 0.13-6), Matrix (>= 1.0-3), digest (>= 0.2.3), matrixStats (>= 0.50.2), png (>= 0.1-2), R.utils (>= 1.32.1), stringi (>= 0.2-3), stringr (>= 0.4), XML (>= 2.1-0), signal, boot, lme4, lifecycle

Suggests Cairo (>= 1.5-0), dichromat (>= 1.2-3), foreach, forecast (>= 3.6), gmp (>= 0.5-5), iterators, knitr, RColorBrewer, rmarkdown, testthat (>= 0.8), tikzDevice, waveslim

Description Perform tree-ring analyses such as detrending, chronology building, and cross dating. Read and write standard file formats used in dendrochronology.

LazyData no

License GPL (>= 2)

URL https://github.com/OpenDendro/dplR

NeedsCompilation yes

Author Andy Bunn [aut, cph, cre, trl],
Mikko Korpela [aut, cph, trl],
Franco Biondi [aut, cph],
Filipe Campelo [aut, cph],
Stefan Klesse [aut, cph],
Pierre Mérian [aut, cph],
Fares Qeadan [aut, cph],
Christian Zang [aut, cph],
Allan Buras [ctb],
Alice Cecile [ctb],
Manfred Mudelsee [ctb],
Michael Schulz [ctb],

David Frank [ctb],
Ronald Visser [ctb],
Ed Cook [ctb],
Kevin Anchukaitis [ctb]

Maintainer Andy Bunn <bunna@wwu.edu>

Repository CRAN

Date/Publication 2025-01-30 23:00:02 UTC

Contents

`dplR-package` *Dendrochronology Program Library in* R

Description

This package contains functions for performing some standard tree-ring analyses.

Details

Package:	dplR
Type:	Package
License:	GPL (>= 2)

Main Functions

`read.rwl` reads rwl files

`detrend` detrends raw ring widths

`chron` builds chronologies

`corr.rwl.seg` crossdating function

Author(s)

Andy Bunn <andy.bunn@wwu.edu> with major additions from Mikko Korpela and other significant contributions from Franco Biondi, Filipe Campelo, Pierre Mérian, Fares Qeadan and Christian Zang. Function `redfit` is an improved translation of program REDFIT which is original work of Manfred Mudelsee and Michael Schulz. Jacob Cecile contributed a bug fix to `detrend.series`. Allan Buras came up with the revised formula of `glk` in dplR >= 1.6.1.

References

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Fritts, H. C. (2001) *Tree Rings and Climate*. Blackburn. ISBN-13: 978-1-930665-39-2.

ads *Age-Dependent Spline*

Description

Applies an age-dependent smoothing spline to *y*.

Usage

```
ads(y, nyrs0 = 50, pos.slope = TRUE)
```

Arguments

y	a numeric vector, typically a tree-ring series.
nyrs0	a number greater than one, affecting the rigidity of the initial spline. A larger *nyrs0* produces a more rigid spline.
pos.slope	a logical flag. Will allow for a positive slope at the end of fitted value. If FALSE the line will be horizontal. See details.

Details

This implements the age-dependent smoothing spline similar to that described by Melvin (2004). In this implementation a cubic smoothing spline (caps) is fit to *y* with an initial stiffness of *nyrs0*. For each succesive measurement, the siffness is incremented by that ring index. This results in a spline is *nyrs0* flexible at the start of the series and grows progressively stiffer. This can help capture the initial fast growth of a juvinielle tree with a flexible spline that then progresses to a stiffer spline that can better model the constant growth commonly found in mature trees. In its details, the cubic smoothing spline follows the Cook and Peters (1981) spline with a 50% frequency cutoff. See Cook and Kairiukstis (1990) for more information.

The default setting for *nyrs0* is 50 years which is approprite for trees with a classic growth model but a value of 10 or 20 might be more appropriate for a Hugershoff-like initial increase in growth. Cook (pers comm) suggests a value of 20 for RCS.

If *pos.slope* is TRUE, the function will attempt to prevent a postive slope at the end of the series. In some cases when *ads* is used for detrending, a postive slope can be considered considered biologically unlikely. In those cases, the user can contrain the positive slope in the slpine. This works by calculating the spline and taking the first difference. Then the function finds the last (outside) index where the spline changes slope and fixes the spline values from the that point to the end. Finally the spline is rerun along this contrained curve. See examples for details. The wisdom of contraining the slope in this manner depends very much on expert knowledge of the system.

Value

A filtered vector.

Author(s)

Fortran code provided by Ed Cook. Ported and adapted for dplR by Andy Bunn.

References

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Melvin, T. M. (2004) *Historical Growth Rates and Changing Climatic Sensitivity of Boreal Conifers*. PhD Thesis, Climatic Research Unit, School of Environmental Sciences, University of East Anglia.

Cook, E. R. and Peters, K. (1981) The Smoothing Spline: A New Approach to Standardizing Forest Interior Tree-Ring Width Series for Dendroclimatic Studies. Tree-Ring Bulletin, 41, 45-53.

See Also

caps, detrend

Examples

```
# fit a curve
data(co021)
aSeries <- na.omit(co021$`641114`)
plot(aSeries,type="l",col="grey50")
lines(ads(y = aSeries),col="blue",lwd=2)

# show an artificial series with a Hugershoff-like curve.
a <- 0.5
b <- 1
g <- 0.1
d <- 0.25

n <- 300
x <- 1:n
y <- I(a*x^b*exp(-g*x)+d)
# add some noise
y <- y + runif(n=length(y),min = 0,max = 0.5)
# Plot with two different splines.
plot(y,type="l",col="grey50")
lines(ads(y,50),col="darkgreen",lwd=2) # bad
lines(ads(y,10),col="darkblue",lwd=2) # good

# now repeat with a positive slope to constrain
y <- I(a*x^b*exp(-g*x)+d)
y[251:300] <- y[251:300] + seq(0,0.25,length.out=50)
y <- y + runif(n=length(y),min = 0,max = 0.5)
plot(y,type="l",col="grey50")
lines(ads(y,10),col="darkgreen",lwd=2) #bad?
lines(ads(y,10,pos.slope=FALSE),col="darkgreen",lwd=2,lty="dashed")
```

---

anos1 *Rothenburg Tree Ring Widths*

Description

This data set gives the raw ring widths for Norway spruce *Picea abies* at Rothenburg ob der Tauber, Bavaria, Germany. There are 20 series from 10 trees. Data set was created using `read.rwl` and saved to an .rda file using `save`.

Usage

```
data(anos1)
```

Format

A `data.frame` containing 20 tree-ring series from 10 trees in columns and 98 years in rows. The correct stc mask for use with `read.ids` is `c(5, 2, 1)`.

References

Zang, C. (2010) *Growth reaction of temperate forest tree species to summer drought – a multispecies tree-ring network approach*. Ph.D. thesis, Technische Universität München.

`as.rwl` *as.rwl*

Description

Attempts to turn its argument into a rwl object.

Usage

```
as.rwl(x)
```

Arguments

`x`	a `data.frame` or `matrix` with series as columns and years as rows

Details

This tries to coerce `x` into class `c("rwl","data.frame")`. Failable.

Value

An object of class `c("rwl", "data.frame")` with the series in columns and the years as rows. The series IDs are the column names and the years are the row names.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

Examples

```
library(graphics)
library(stats)
library(utils)
## Toy
n <- 100
## Make a data.frame that is tree-ring like
base.series <- 0.75 + exp(-0.2 * 1:n)
foo <- data.frame(x1 = base.series + abs(rnorm(n, 0, 0.25)),
                  x2 = base.series + abs(rnorm(n, 0, 0.25)),
                  x3 = base.series + abs(rnorm(n, 0, 0.25)),
                  x4 = base.series + abs(rnorm(n, 0, 0.25)),
                  x5 = base.series + abs(rnorm(n, 0, 0.25)),
                  x6 = base.series + abs(rnorm(n, 0, 0.25)))
# coerce to rwl and use plot and summary methods
foo <- as.rwl(foo)
class(foo)
plot(foo, plot.type="spag")
summary(foo)
```

---

bai.in *Basal Area Increment (Inside Out)*

Description

Convert multiple ring-width series to basal area increment (i.e., ring area) going from the pith to the bark.

Usage

```
bai.in(rwl, d2pith = NULL)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl
d2pith	an optional data.frame containing two variables. If present, then variable one (series in the example below) gives the series ID as either characters or factors. These must exactly match colnames(*rwl*). Variable two (*d2pith* in the example below) gives the distance from the innermost measured ring to the pith of the tree in mm. If *d2pith* is NULL then the distance to pith is assumed to be zero for each series (column) in *rwl*.

Details

This converts ring-width series (mm) to ring-area series (mm squared) (aka basal area increments) based on the distance between the innermost measured ring and the pith of the tree. It is related to bai.out, which calculates each ring area starting from the outside of the tree and working inward. Both methods assume a circular cross section (Biondi 1999). See the references below for further details.

Value

A `data.frame` containing the ring areas for each series with column names, row names and dimensions of *rwl*.

Note

DendroLab website: https://dendrolaborg.wordpress.com/

Author(s)

Code by Andy Bunn based on work from DendroLab, University of Nevada Reno, USA. Patched and improved by Mikko Korpela.

References

Biondi, F. (1999) Comparing tree-ring chronologies and repeated timber inventories as forest monitoring tools. *Ecological Applications*, **9**(1), 216–227.

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

See Also

`bai.out`

Examples

```
library(graphics)
library(stats)
library(utils)
## Toy
n <- 100
## Make three fake tree-ring series to show that these funcs work on rwl objects
base.series <- 0.75 + exp(-0.2 * 1:n)
rwl <- data.frame(x1 = base.series + abs(rnorm(n, 0, 0.05)),
                  x2 = base.series + abs(rnorm(n, 0, 0.05)),
                  x3 = base.series + abs(rnorm(n, 0, 0.05)))

## The inside out method
foo <- bai.in(rwl = rwl)
## The outside in method
bar <- bai.out(rwl = rwl)

## Identical
head(bar)
head(foo)

## Use gp data
data(gp.rwl)
data(gp.d2pith)
foo <- bai.in(rwl = gp.rwl, d2pith = gp.d2pith)
foo.crn <- chron(foo)
```

```
yrs <- time(foo.crn)
plot(yrs, foo.crn[, 1], type = "n",
     xlab = "Year", ylab = expression(mm^2))
lines(yrs, foo.crn[, 1], col = "grey", lty = "dashed")
lines(yrs, caps(foo.crn[, 1], nyrs = 32), col = "red", lwd = 2)
```

`bai.out` *Basal Area Increment (Outside In)*

Description

Convert multiple ring-width series to basal area increment (i.e., ring area) going from the bark to the pith.

Usage

```
bai.out(rwl, diam = NULL)
```

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`
`diam`	an optional `data.frame` containing two variables. If present, variable one (*series* in the example below) gives the series ID as either `characters` or `factors`. These must exactly match `colnames(`*rwl*`)`. Variable two (*diam* in the example below) gives the diameter of the tree (in mm) up to the outermost ring (e.g., the diameter of the tree where the core was taken minus the thickness of the bark). If *diam* is `NULL` then the diameter is taken as twice the sum of the widths for each series (column) in *rwl*.

Details

This converts ring-width series (mm) to ring-area series (mm squared) (aka basal area increments) based on the diameter of the tree and the width of each ring moving towards the pith of the tree. It is related to `bai.in`, which calculates each ring area starting from the inside of the tree and working outward. Both methods assume a circular cross section (Biondi 1999). See the references below for further details.

Value

A `data.frame` containing the ring areas for each series with column names, row names and dimensions of *rwl*.

Note

DendroLab website: https://dendrolaborg.wordpress.com/

Author(s)

Code by Andy Bunn based on work from DendroLab, University of Nevada Reno, USA. Patched and improved by Mikko Korpela.

References

Biondi, F. (1999) Comparing tree-ring chronologies and repeated timber inventories as forest monitoring tools. *Ecological Applications*, **9**(1), 216–227.

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

See Also

bai.in

Examples

```
library(graphics)
library(utils)
## Not run:
library(stats)
## Toy
n <- 100
## Make three fake tree-ring series to show that these funcs work on rwl objects
base.series <- 0.75 + exp(-0.2 * 1:n)
rwl <- data.frame(x1 = base.series + abs(rnorm(n, 0, 0.05)),
                  x2 = base.series + abs(rnorm(n, 0, 0.05)),
                  x3 = base.series + abs(rnorm(n, 0, 0.05)))

## The inside out method
foo <- bai.in(rwl = rwl)
## The outside in method
bar <- bai.out(rwl = rwl)

## Identical
head(bar)
head(foo)

## End(Not run)
## Use gp data
data(gp.rwl)
data(gp.dbh)
## dbh (minus the bark) from cm to mm
gp.dbh2 <- gp.dbh[, 1:2]
gp.dbh2[, 2] <- (gp.dbh[, 2] - gp.dbh[, 3]) * 10
bar <- bai.out(rwl = gp.rwl, diam = gp.dbh2)
bar.crn <- chron(bar)
yrs <- time(bar.crn)
plot(yrs, bar.crn[, 1], type = "n",
     xlab = "Year", ylab = expression(mm^2))
lines(yrs, bar.crn[, 1], col = "grey", lty = "dashed")
```

```
lines(yrs, caps(bar.crn[, 1], nyrs = 32), col = "red", lwd = 2)
```

---

bakker *Basal Area Increment (Bakker)*

Description

Convert multiple ring-width series to basal area increment (i.e., ring area) following the proportional method of Bakker (2005).

Usage

```
bakker(rwl, ancillary)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl
ancillary	A data.frame containing four columns. Column one (series in the example below) gives the series ID as either characters or factors. These must exactly match colnames(rwl). Column two (PO) gives the number of rings estimated to be missing to the pith. Column three (d2pith) gives the estimated distance to the pith (mm). Column four (diam) gives the diameter at breast height (DBH) in cm.

Details

This converts ring-width series (mm) to ring-area series (mm squared) (aka basal area increments) based on the diameter of the tree, the missing distance to the pith and the missing number of rings to the pith, following the proportional method for reconstructing historical tree diameters by Bakker (2005). It prevents bai.out transformations from producing negative increments when the sum of all ring widths in a series is larger than DBH/2. It prevents bai.in transformations from producing too small values when the sum of all ring widths in a series is smaller than DBH/2.

Value

A list containing the following objects:

DBHhist_raw	data.frame with the reconstructed diameter for each series with column names, row names and dimensions of rwl.
baiBakker_raw	data.frame with the basal area increments for each series with column names, row names and dimensions of rwl.

Author(s)

Code by Stefan Klesse. Adapted for dplR by Andy Bunn.

References

Bakker, J.D., 2005. A new, proportional method for reconstructing historical tree diameters. Canadian Journal of Forest Research 35, 2515–2520. https://doi.org/10.1139/x05-136

Examples

```
data(zof.rwl)
data(zof.anc)
zof.bakker <- bakker(rwl = zof.rwl,ancillary = zof.anc)
zof.bai <- zof.bakker$baiBakker_raw

# first series bai
yrs <- time(zof.rwl)
plot(yrs,zof.bai[,1],type="l",
     xlab="Year",
     ylab=expression(BAI~(mm^2)),
     main = colnames(zof.bai)[1])
```

---

ca533 *Campito Mountain Tree Ring Widths*

Description

This data set gives the raw ring widths for bristlecone pine *Pinus longaeva* at Campito Mountain in California, USA. There are 34 series. Data set was created using `read.rwl` and saved to an .rda file using `save`.

Usage

```
data(ca533)
```

Format

A `data.frame` containing 34 tree-ring series in columns and 1358 years in rows.

Source

International tree-ring data bank, Accessed on 20-April-2021 at https://www.ncei.noaa.gov/pub/data/paleo/treering/measurements/northamerica/usa/ca533.rwl

References

Graybill, D. A. and LaMarche, Jr., V. C. (1983) Campito Mountain Data Set. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series 1983-CA533.RWL. NOAA/NCDC Paleoclimatology Program, Boulder, Colorado, USA.

cana157 *Twisted Tree Heartrot Hill Standard Chronology*

Description

This data set gives the standard chronology for white spruce *Picea glauca* at Twisted Tree Heartrot Hill in Yukon, Canada. Data set was created using `read.crn` and saved to an .rda file using `save`.

Usage

```
data(cana157)
```

Format

A `data.frame` containing the standard chronology in column one and the sample depth in column two. There are 463 years (1530–1992) in the rows.

Source

International tree-ring data bank, Accessed on 20-April-2021 at https://www.ncei.noaa.gov/pub/data/paleo/treering/chronologies/northamerica/canada/cana157.crn

References

Jacoby, G., D'Arrigo, R. and Buckley, B. (1992) Twisted Tree Heartrot Hill Data Set. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series 1992-CANA157.CRN. NOAA/NCDC Paleoclimatology Program, Boulder, Colorado, USA.

caps *Cook and Peters Smoothing Spline with User-Specified Rigidity and Frequency Cutoff*

Description

Applies a smoothing spline to *y* with rigidity determined by two parameters: frequency response *f* at a wavelength of *nyrs* years.

Usage

```
caps(y, nyrs = 32, f = 0.5)
```

Arguments

y	a numeric vector, typically a tree-ring series.
nyrs	a number greater than zero, affecting the rigidity of the spline. If nyrs is between zero and one it will be treated as percentage of the length of the series. E.g., a value of 0.667 would 0.667 * length(y) or what is often called a 2/3 spline in the tree-ring literature. When *f* is kept constant, a larger *nyrs* produces a more rigid spline.
f	a number between 0 and 1 giving the frequency response at a wavelength of *nyrs* years. When *nyrs* is kept constant, a smaller *f* produces a more rigid spline: At one extreme, *f* = 0 causes the function to return the least-squares straight line fit to the data. At the other extreme, as *f* approaches 1 the result approaches the natural spline, i.e. the function outputs *y*. The default value is 0.5 and shouldn't be changed without a good reason.

Details

This applies the classic smoothing spline from Cook and Peters (1981). The rigidity of the spline has a frequency response of 50% at a wavelength of *nyrs*. The references, of course, have more information.

This function was introduced to *dplR* in version 1.7.3 and replaces the now defunct *ffcsaps*. Where *ffcsaps* was written entirely in R, *caps* is a wrapper for a Fortran subroutine from Ed Cook's ARSTAN program that is thousands of times faster.

The default value of *nyrs* was changed to 32 from 2/3 length of *y* in version 1.7.8 based on a suggestion from Klesse (2021).

Note: *caps* returns NA if there are any NA values in y. See examples.

Value

A filtered vector.

Author(s)

Fotran code provided by Ed Cook and adapted for dplR by Andy Bunn.

References

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Cook, E. R. and Peters, K. (1981) The Smoothing Spline: A New Approach to Standardizing Forest Interior Tree-Ring Width Series for Dendroclimatic Studies. Tree-Ring Bulletin, 41, 45-53.

Klesse, S. (2021) Critical Note on the Application of the "Two-Third"" Spline. Dendrochronologia Volume 65: 125786

See Also

ads

Examples

```
library(graphics)
library(utils)

## Use first series from the Mesa Verde data set
data(co021)
series <- co021[, 1]
series <- series[!is.na(series)]
plot(series, type = "l", ylab = "Ring Width (mm)", col = "grey")
lines(caps(series, nyrs = 10), col = "red", lwd = 2)
lines(caps(series, nyrs = 100), col = "green", lwd = 2)
# A 2/3 spline, the default, 462.6667 yrs here
lines(caps(series), col = "blue", lwd = 2)
legend("topright",
       c("Series", "nyrs=10", "nyrs=100",
         paste("Default nyrs (", floor(length(series) * 2/3), ")", sep="")),
       fill=c("grey", "red", "green", "blue"))


## Note behavior when NA is encountered and
## take appropriate measures as demonstrated above
y <- c(NA,NA,rnorm(100))
caps(y)
```

`ccf.series.rwl` *Cross-Correlation between a Series and a Master Chronology*

Description

Computes cross-correlations between a tree-ring series and a master chronology built from a rwl object at user-specified lags and segments.

Usage

```
ccf.series.rwl(rwl, series, series.yrs = as.numeric(names(series)),
               seg.length = 50, bin.floor = 100, n = NULL,
               prewhiten = TRUE, biweight = TRUE, pcrit = 0.05,
               lag.max = 5, make.plot = TRUE,
               floor.plus1 = FALSE, series.x = FALSE, ...)
```

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`.
`series`	a `numeric` or `character` vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be `numeric`.

series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)). Ignored if *series* is an index to a column of *rwl*.
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
bin.floor	a non-negative integral value giving the base for locating the first segment (e.g., 1600, 1700, 1800 AD). Typically 0, 10, 50, 100, etc.
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
prewhiten	logical flag. If TRUE each series is whitened using ar.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
pcrit	a number between 0 and 1 giving the critical value for the correlation test.
lag.max	an integral value giving the maximum lag at which to calculate the ccf.
make.plot	logical flag indicating whether to make a plot.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
series.x	logical flag indicating whether to make the series the x argument to ccf. See Details.
...	other arguments passed to plot.

Details

This function calculates the cross-correlation function between a tree-ring series and a master chronology built from *rwl* looking at correlations lagged positively and negatively using ccf at overlapping segments set by *seg.length*. For instance, with *lag.max* set to 5, cross-correlations would be calculated at for each segment with the master lagged at k = -5:5 years.

The cross correlations are calculated calling ccf as ccf(x=master, y=series, lag.max=lag.max, plot=FALSE) if series.x is FALSE and as ccf(x=series, y=master, lag.max=lag.max, plot=FALSE) if series.x is TRUE. This argument was introduced in dplR version 1.7.0. Different users have different expectations about how missing or extra rings are notated. If switch.x = FALSE the behavior will be like COFECHA where a missing ring in a series produces a negative lag in the plot rather than a positive lag.

Correlations are calculated for the first segment, then the second segment and so on. Correlations are only calculated for segments with complete overlap with the master chronology.

Each series (including those in the *rwl* object) is optionally detrended as the residuals from a hanning filter with weight *n*. The filter is not applied if *n* is NULL. Detrending can also be done via prewhitening where the residuals of an ar model are added to each series mean. This is the default. The master chronology is computed as the mean of the *rwl* object using tbrm if *biweight* is TRUE and rowMeans if not. Note that detrending typically changes the length of the series. E.g., a hanning filter will shorten the series on either end by floor(*n*/2). The prewhitening default will change the series length based on the ar model fit. The effects of detrending can be seen with series.rwl.plot.

Value

A `list` containing matrices *ccf* and *bins*. Matrix *ccf* contains the correlations between the series and the master chronology at the lags window given by *lag.max*. Matrix *bins* contains the years encapsulated by each bin.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Bunn, A. G. (2010) Statistical and visual crossdating in R using the dplR library. *Dendrochronologia*, **28**(4), 251–258.

See Also

`corr.rwl.seg`, `corr.series.seg`, `skel.plot`, `series.rwl.plot`

Examples

```
library(utils)
data(co021)
dat <- co021
## Create a missing ring by deleting a year of growth in a random series
flagged <- dat$"641143"
flagged <- c(NA, flagged[-325])
names(flagged) <- rownames(dat)
dat$"641143" <- NULL
ccf.100 <- ccf.series.rwl(rwl = dat, series = flagged, seg.length = 100)
## Not run:
flagged2 <- co021$"641143"
names(flagged2) <- rownames(dat)
ccf.100.1 <- ccf.series.rwl(rwl = dat, seg.length = 100,
                            series = flagged2)
## Select series by name or column position
ccf.100.2 <- ccf.series.rwl(rwl = co021, seg.length = 100,
                            series = "641143")
ccf.100.3 <- ccf.series.rwl(rwl = co021, seg.length = 100,
                            series = which(colnames(co021) == "641143"))
identical(ccf.100.1, ccf.100.2) # TRUE
identical(ccf.100.2, ccf.100.3) # TRUE

## End(Not run)
```

chron *Build Mean Value Chronology*

Description

This function builds a mean value chronology, typically from a `data.frame` of detrended ring widths as produced by `detrend`.

Usage

```
chron(x, biweight = TRUE, prewhiten = FALSE, ...)
```

Arguments

`x`	a `data.frame` of (usually detrended) ring widths with `rownames(x)` containing years and `colnames(x)` containing each series ID such as produced by `read.rwl`
`biweight`	`logical` flag. If TRUE then a robust mean is calculated using `tbrm`.
`prewhiten`	`logical` flag. If TRUE each series is whitened using `ar` prior to averaging.
`...`	Arguments passed to `ar` when `prewhiten` is `TRUE`. For example, use `aic` and `order.max` to control the order of the AR model.

Details

This either averages the rows of the `data.frame` using a mean or a robust mean (the so-called standard chronology) or can do so from the residuals of an AR process (the residual chronology).

Note that the residual chronology in this function will return different values than the residual chronology from `chron.ars` which uses a slightly different method for determining AR order.

Value

An object of of class `crn` and `data.frame` with the standard chronology, residual chronology (if prewhitening was performed), and the sample depth. The years are stored as row numbers.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Fritts, H. C. (2001) *Tree Rings and Climate*. Blackburn. ISBN-13: 978-1-930665-39-2.

See Also

`read.rwl`, `detrend`, `ar`, `crn.plot`

Examples

```
library(graphics)
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi)
plot(ca533.crn,xlab="Year",ylab="RWI")
## With residual chron
ca533.crn2 <- chron(ca533.rwi, prewhiten = TRUE)
plot(ca533.crn2,xlab="Year",ylab="RWI")
```

`chron.ars` *Build ARSTAN Chronology*

Description

This function builds three varieties of the mean-value chronology, including the ARSTAN chronology, typically from a `data.frame` of detrended ring widths as produced by `detrend`.

Usage

```
chron.ars(x, biweight=TRUE, maxLag=10, firstAICmin=TRUE,
  verbose=TRUE, prewhitenMethod=c("ar.yw","arima.CSS-ML"))
```

Arguments

`x`	a `data.frame` of (usually detrended) ring widths with `rownames(x)` containing years and `colnames(x)` containing each series ID such as produced by `read.rwl`
`biweight`	`logical` flag. If TRUE then a robust mean is calculated using `tbrm`.
`maxLag`	an `integer` giving the maximum lag to consider in the AR pooling.
`firstAICmin`	`logical` flag. If TRUE the final AR order is elected using the first AIC minimum otherwise the order is selected by the overall minimum.
`verbose`	`logical` flag. If TRUE the function prints information from the AR modeling to the screen.
`prewhitenMethod`	a `character` vector to determine the AR model fitting. See details below. Possible values are either "ar.yw" or "arima.CSS-ML". Can be abbreviated. Defaults to "ar.yw".

Details

This produces three mean-value chronologies: standard, residual, and ARSTAN. Users unfamiliar with the concept behind the ARSTAN method should look to Cook (1985) for background and inspiration.

The standard chronology is the (biweight) mean value across rows and identical to `chron`.

The residual chronology is the prewhitened chronology as described by Cook (1985) and uses uses multivariate autoregressive modeling to determine the order of the AR process. It's important to note that residual chronology produced here is different than the simple residual chronology produced by chron which returns the residuals of an AR process using a naive call to ar. But in practice the results will be similar. For more on the residual chronology in this function, see pp. 153-154 in Cook's 1985 dissertation.

The ARSTAN chronology builds on the residual chronology but returns a re-whitened chronology where the pooled AR coefficients from the multivariate autoregressive modeling are reintroduced. See references for details.

The order of the AR model is selected from the pooled AR coefficients by AIC using either the first (local) AIC minimum otherwise or the overall minimum considering the maximum lag (argument maxLag).

Once the AR order is determined an AR(p) model is fit using either ar via the Yule-Walker method or by arima via conditional-sum-of-squares to find starting values, then maximum likelihood. It is possible that the model will not converge in which case a warning is produced. The AR fitting is determined via prewhitenMethod and defaults to using ar.

Value

A data.frame with the standard, residual, and ARSTAN chronologies. The sample depth is also included.

Author(s)

Andy Bunn with contributions from Kevin Achukaitis and Ed Cook. Much of the function is a port of Cook's FORTRAN code.

References

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Cook, E. R. (1985). A Time Series Analysis Approach to Tree Ring Standardization. PhD thesis, The University of Arizona.

See Also

chron, crn.plot, ar, arima

Examples

```
library(graphics)
library(utils)
data(co021)
co021.rwi <- detrend(rwl = co021, method = "AgeDepSpline")
co021.crn <- chron.ars(co021.rwi)
plot(co021.crn,xlab="Year",ylab="RWI",add.spline=TRUE,nyrs=20)
cor(co021.crn)
```

chron.ci *Build Mean Value Chronology with Confidence Intervals*

Description

This function builds a mean value chronology with bootstrapped confidence intervals, typically from a data.frame of detrended ring widths as produced by detrend.

Usage

```
chron.ci(x, biweight=TRUE, conf=0.95, R=100)
```

Arguments

x	a data.frame of (usually detrended) ring widths with rownames(x) containing years and colnames(x) containing each series ID such as produced by read.rwl
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
conf	numeric. A scalar for the confidence level.
R	integer. The number of bootstrap replicates.

Details

This either averages the rows of the data.frame using a mean or a robust mean (the so-called standard chronology) and calculates boostrapped confidence intervals using the normal approximation. The function will fail if there are any rows in x that contain only one sample and in practice there should be several samples in a row. One of the guiding principles of bootstrapping is that the population is to the sample as the sample is to the bootstrap samples.

Value

An object of of class data.frame with the standard chronology, the upper and lower confidence interval, and the sample depth. The years are stored as row numbers.

Author(s)

Andy Bunn.

See Also

read.rwl, detrend, boot, boot.ci

Examples

```
library(graphics)
library(utils)
data(wa082)
# truncate to a sample depth of five
wa082Trunc <- wa082[rowSums(!is.na(wa082))>4,]
# detrend
wa082RWI <- detrend(wa082Trunc, method="AgeDepSpline")
# bootstrap the chronology and
wa082Crn <- chron.ci(wa082RWI, biweight = TRUE, R = 100, conf = 0.99)
head(wa082Crn)
# plot (this is so much easier in ggplot!)
wa082Crn$yrs <- time(wa082Crn)
xx <- c(wa082Crn$yrs,rev(wa082Crn$yrs))
yy <- c(wa082Crn$lowerCI,rev(wa082Crn$upperCI))
plot(wa082Crn$yrs,wa082Crn$std,type="n",ylim=range(yy),
     ylab="RWI",xlab="Year",main="Chronology with CI")
polygon(x=xx,y=yy,col = "grey",border = NA)
lines(wa082Crn$yrs,wa082Crn$std)
```

`chron.stabilized` *Build Mean Value Chronology with Stabilized Variance*

Description

This function builds a variance stabilized mean-value chronology, typically from a `data.frame` of detrended ring widths as produced by `detrend`.

Usage

```
chron.stabilized(x, winLength, biweight = TRUE, running.rbar = FALSE)
```

Arguments

`x`	a `data.frame` of ring widths with `rownames(x)` containing years and `colnames(x)` containing each series ID such as produced by `read.rwl`
`winLength`	a odd `integer` specifying the window length.
`biweight`	`logical` flag. If TRUE then a robust mean is calculated using `tbrm`.
`running.rbar`	`logical` flag. If TRUE the running interseries correlation is returned as well.

Details

The variance of a mean chronology depends on the variance of the individual samples, the number of series averaged together, and their interseries correlation (Wigley et al. 1984). As the number of series commonly decreases towards the beginning of a chronology averaging introduces changes in variance that are a solely an effect of changes in sample depth.

Additionally, time-dependent changes in interseries correlation can cause artificial variance changes of the final mean chronology. The function `chron.stabilized` accounts for both temporal changes in the interseries correlation and sample depth to produce a mean value chronology with stabilized variance.

The basic correction centers around the use of the effective independent sample size, `Neff`, which considers sample replication and mean interseries correlation between the samples at every time. This is defined as: `Neff = n(t) / 1+(n(t)-1)rbar(t)`

where `n(t)` is the number of series at time `t`, and `rbar` is the interseries correlation (see `interseries.cor`). Multiplication of the mean time series with the square root of `Neff` at every time `t` theoretically results in variance that is independent of sample size. In the limiting cases, when the `rbar` is zero or unity, `Neff` obtains values of the true sample size and unity, respectively.

Value

An object of of class `crn` and `data.frame` with the variance stabilized chronology, running interseries correlation ('if `running.bar=TRUE`), and the sample depth.

Author(s)

Original code by David Frank and adapted for dplR by Stefan Klesse. Patched and improved by Andy Bunn.

References

Frank, D, Esper, J, Cook, E, (2006) *On variance adjustments in tree-ring chronology development*. Tree rings in archaeology, climatology and ecology, TRACE 4, 56–66

Frank, D, Esper, J, Cook, E, (2007) *Adjustment for proxy number and coherence in a large-scale temperature reconstruction*. Geophysical Research Letters 34

Wigley, T, Briffa K, Jones P (1984) *On the Average Value of Correlated Time Series, with Applications in Dendroclimatology and Hydrometeorology*. J. Climate Appl. Meteor., 23, 201–213

See Also

`chron`

Examples

```
library(graphics)
library(utils)
data(co021)
co021.rwi <- detrend(co021,method = "Spline")
co021.crn <- chron(co021.rwi)
co021.crn2 <- chron.stabilized(co021.rwi,
                               winLength=101,
                               biweight = TRUE,
                               running.rbar = FALSE)
yrs <- time(co021)
plot(yrs,co021.crn$std,type="l",col="grey")
lines(yrs,co021.crn2$adj.crn,col="red")
```

cms *C-Method Standardization*

Description

Detrend multiple ring-width series simultaneously using the C-method.

Usage

```
cms(rwl, po, c.hat.t = FALSE, c.hat.i = FALSE)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl
po	a data.frame containing two variables. Variable one (*series* in the example below) gives the series ID as either characters or factors. These must exactly match colnames(*rwl*). Variable two (*pith.offset* in the example below) must be integral values and give the years from the beginning of the core to the pith (or center) of the tree.
c.hat.t	a logical indicating whether to export the C-curves for each tree by biological age.
c.hat.i	a logical indicating whether to export the expected ring widths for each series.

Details

This method detrends and standardizes tree-ring series by calculating a growth curve based on constant annual basal area increment. The method is based on the "assumption that constant growth is expressed by a constant basal area increment distributed over a growing surface" (Biondi and Qeadan 2008). The detrending is the estimation and removal of the tree's natural biological growth trend. The standardization is done by dividing each series by the growth trend to produce units in the dimensionless ring-width index (RWI).

This attempts to remove the low frequency variability that is due to biological or stand effects.

See the reference below for further details.

Value

A data.frame containing the dimensionless and detrended ring-width indices with column names, row names and dimensions of *rwl* if *c.hat.t* is FALSE and *c.hat.i* is FALSE.

Otherwise a list of length 2 or 3 containing the RWI data.frame, a data.frame containing the C-curves for each tree (*c.hat.t*), and/or a vector containing the C-values for each tree (*c.hat.i*) depending on the output flags. See Eq. 12 in Biondi and Qeadan (2008) for more detail on *c.hat.t*, and *c.hat.i*.

Note

DendroLab website: https://dendrolaborg.wordpress.com/

Author(s)

Code provided by DendroLab based on programming by F. Qeadan and F. Biondi, University of Nevada Reno, USA and adapted for dplR by Andy Bunn. Patched and improved by Mikko Korpela.

References

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

See Also

detrend, chron, rcs

Examples

```
library(graphics)
library(utils)
data(gp.rwl)
data(gp.po)
gp.rwi <- cms(rwl = gp.rwl, po = gp.po)
gp.crn <- chron(gp.rwi)
crn.plot(gp.crn, add.spline = TRUE)
## c.hat
gp.rwi <- cms(rwl = gp.rwl, po = gp.po, c.hat.t = TRUE, c.hat.i = TRUE)
dotchart(gp.rwi$c.hat.i, ylab = "Series", xlab = expression(hat(c)[i]))
tmp <- gp.rwi$c.hat.t
plot(tmp[, 1], type = "n", ylim = range(tmp, na.rm = TRUE),
     xlab = "Cambial Age", ylab = expression(hat(c)[t]))
apply(tmp, 2, lines)
```

---

co021 *Schulman Old Tree No. 1, Mesa Verde*

Description

This data set gives the raw ring widths for Douglas fir *Pseudotsuga menziesii* at Mesa Verde in Colorado, USA. There are 35 series. Data set was created using read.rwl and saved to an .rda file using save.

Usage

```
data(co021)
```

Format

A data.frame containing 35 tree-ring series in columns and 788 years in rows.

Source

International tree-ring data bank, Accessed on 20-April-2021 at https://www.ncei.noaa.gov/pub/data/paleo/treering/measurements/northamerica/usa/co021.rwl

References

Schulman, E. (1963) Schulman Old Tree No. 1 Data Set. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series 1983-CO021.RWL. NOAA/NCDC Paleoclimatology Program, Boulder, Colorado, USA.

---

combine.rwl *Combine Tree-Ring Data Sets*

Description

This function combines any number of data.frames of tree-ring data into one data.frame.

Usage

```
combine.rwl(x, y = NULL)
```

Arguments

x	either a data.frame to be combined with *y*, or a list of data.frames to be combined.
y	a data.frame to be combined with data.frame *x*.

Details

The sequence of years in each data.frame must be increasing and continuous. The output produced by the function also fulfills this condition. If the input is differently formatted, the result will be wrong.

Value

An object of class c("rwl", "data.frame") with the series in columns and the years as rows. The keycodes are the column names and the years are the row names.

Author(s)

Christian Zang. Patched by Mikko Korpela.

Examples

```
library(utils)
data(ca533)
data(co021)
combi1 <- combine.rwl(list(ca533, co021))
## or alternatively for data.frames to combine
combi2 <- combine.rwl(ca533, co021)
identical(combi1, combi2) # TRUE
```

`common.interval` *Common Interval*

Description

This function finds the common interval on a set of tree-ring widths such as that produced by `read.rwl`.

Usage

```
common.interval(rwl, type=c("series", "years", "both"),
                make.plot=TRUE)
```

Arguments

`rwl`	a `data.frame` of ring widths with `rownames(x)` containing years and `colnames(x)` containing each series ID such as produced by `read.rwl`
`type`	a `character` string of `"series"`, `"years"`, or `"both"`. Argument matching is performed.
`make.plot`	a `logical` indicating if a plot should be drawn

Details

This trims an `rwl` object to a common interval that maximizes the number of series (`type="series"`), the number of years (`type="years"`), or a compromise between the two (`type="both"`). A modified `seg.plot` can be drawn as well.

Value

A `data.frame` with `colnames(x)` and `rownames(x)`.

Author(s)

Filipe Campelo, Andy Bunn and Mikko Korpela

See Also

`seg.plot`

Examples

```
library(utils)
data(co021)
co021.s <- common.interval(co021, type="series", make.plot=TRUE)
co021.y <- common.interval(co021, type="years", make.plot=TRUE)
co021.b <- common.interval(co021, type="both", make.plot=TRUE)

dim(co021)
dim.s <- dim(co021.s)
dim.s       # the highest number of series
prod(dim.s) #   (33 series x 288 years = 9504)
dim.y <- dim(co021.y)
dim.y       # the highest number of years
prod(dim.y) #   (27 series x 458 years = 12366)
dim.b <- dim(co021.b)
dim.b       # compromise solution
prod(dim.b) #   (28 series x 435 years = 12180)
```

---

corr.rwl.seg *Compute Correlations between Series*

Description

Computes the correlation between each tree-ring series in a rwl object.

Usage

```
corr.rwl.seg(rwl, seg.length = 50, bin.floor = 100, n = NULL,
             prewhiten = TRUE, pcrit = 0.05, biweight = TRUE,
             method = c("spearman", "pearson","kendall"),
             make.plot = TRUE, label.cex = 1, floor.plus1 = FALSE,
             master = NULL,
             master.yrs = as.numeric(if (is.null(dim(master))) {
                              names(master)
                          } else {
                              rownames(master)
                          }),
             ...)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
bin.floor	a non-negative integral value giving the base for locating the first segment (e.g., 1600, 1700, 1800 AD). Typically 0, 10, 50, 100, etc.

n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
prewhiten	logical flag. If TRUE each series is whitened using ar.
pcrit	a number between 0 and 1 giving the critical value for the correlation test.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
method	Can be either "pearson", "kendall", or "spearman" which indicates the correlation coefficient to be used. Defaults to "spearman". See cor.test.
make.plot	logical flag indicating whether to make a plot.
label.cex	numeric scalar for the series labels on the plot. Passed to *axis.cex* in axis.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
master	NULL, a numeric vector or a matrix-like object of numeric values, including a data.frame. If NULL, a number of master chronologies, one for each series in *rwl*, is built from *rwl* using the leave-one-out principle. If a vector, the function uses this as the master chronology. If a matrix or data.frame, this object is used for building the master chronology (no leave-one-out).
master.yrs	a numeric vector giving the years of *series*. Defaults to names or rownames of *master* coerced to numeric type.
...	other arguments passed to plot.

Details

This function calculates correlation serially between each tree-ring series and a master chronology built from all the other series in the *rwl* object (leave-one-out principle). Optionally, the user may give a *master* chronology (a vector) as an argument. In the latter case, the same master chronology is used for all the series in the *rwl* object. The user can also choose to give a *master* data.frame (series as columns, years as rows), from which a single master chronology is built.

Correlations are done for each segment of the series where segments are lagged by half the segment length (e.g., 100-year segments would be overlapped by 50-years). The first segment is placed according to *bin.floor*. The minimum bin year is calculated as ceiling(*min.yr*/*bin.floor*)∗*bin.floor* where *min.yr* is the first year in either the *rwl* object or the user-specified *master* chronology, whichever is smaller. For example if the first year is 626 and *bin.floor* is 100 then the first bin would start in 700. If *bin.floor* is 10 then the first bin would start in 630.

Correlations are calculated for the first segment, then the second segment and so on. Correlations are only calculated for segments with complete overlap with the master chronology. For now, correlations are Spearman's rho calculated via cor.test using method = "spearman".

Each series (including those in the rwl object) is optionally detrended as the residuals from a hanning filter with weight *n*. The filter is not applied if *n* is NULL. Detrending can also be done via prewhitening where the residuals of an ar model are added to each series mean. This is the default. The master chronology is computed as the mean of the *rwl* object using tbrm if *biweight* is TRUE and rowMeans if not. Note that detrending can change the length of the series. E.g., a hanning filter will shorten the series on either end by floor(*n*/2). The prewhitening default will change the series length based on the ar model fit. The effects of detrending can be seen with series.rwl.plot.

The function is typically invoked to produce a plot where each segment for each series is colored by its correlation to the master chronology. Green segments are those that do not overlap completely with the width of the bin. Blue segments are those that correlate above the user-specified critical value. Red segments are those that correlate below the user-specified critical value and might indicate a dating problem.

Value

A `list` containing matrices *spearman.rho*, *p.val*, *overall*, *bins*, *rwi*, vector *avg.seg.rho*, numeric *seg.lag*, *seg.length*, *pcrit*, *label.cex*. An additional `character` *flags* is also returned if any segments fall below the critical value. Matrix *spearman.rho* contains the correlations for each series by bin. Matrix *p.val* contains the p-values on the correlation for each series by bin. Matrix *overall* contains the average correlation and p-value for each series. Matrix *bins* contains the years encapsulated by each bin. The vector *avg.seg.rho* contains the average correlation for each bin. Matrix *rwi* contains the detrended rwl data, the numerics *seg.lag*, *seg.length*, *pcrit*, *label.cex* are from the oroginal call and used to pass into *plot.crs*.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

`corr.series.seg`, `skel.plot`, `series.rwl.plot`, `ccf.series.rwl`, `plot.crs`

Examples

```
library(utils)
data(co021)
crs <- corr.rwl.seg(co021, seg.length = 100, label.cex = 1.25)
names(crs)
## Average correlation and p-value for the first few series
head(crs$overall)
## Average correlation for each bin
crs$avg.seg.rho
```

`corr.series.seg` *Compute Correlation between a Series and a Master Chronology*

Description

Compute correlation between a tree-ring series and a master chronology by segment.

Usage

```
corr.series.seg(rwl, series, series.yrs = as.numeric(names(series)),
                seg.length = 50, bin.floor = 100, n = NULL,
                prewhiten = TRUE, biweight = TRUE,
                method = c("spearman", "pearson","kendall"),
                pcrit = 0.05,
                make.plot = TRUE, floor.plus1 = FALSE, ...)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)). Ignored if *series* is an index to a column of *rwl*.
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
bin.floor	a non-negative integral value giving the base for locating the first segment (e.g., 1600, 1700, 1800 AD). Typically 0, 10, 50, 100, etc.
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
prewhiten	logical flag. If TRUE each series is whitened using ar.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
method	Can be either "pearson", "kendall", or "spearman" which indicates the correlation coefficient to be used. Defaults to "spearman". See cor.test.
pcrit	a number between 0 and 1 giving the critical value for the correlation test.
make.plot	logical flag indicating whether to make a plot.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
...	other arguments passed to plot.

Details

This function calculates the correlation between a tree-ring series and a master chronology built from a rwl object. Correlations are done by segment (see below) and with a moving correlation with length equal to the *seg.length*. The function is typically invoked to produce a plot.

Value

A list containing matrices *bins*, *moving.rho*, and vectors *spearman.rho*, *p.val*, and *overall*.

Matrix *bins* contains the years encapsulated by each bin (segments). Matrix *moving.rho* contains the moving correlation and p-value for a moving average equal to *seg.length*. Vector *spearman.rho*

contains the correlations by bin and *p.val* contains the p-values. Vector *overall* contains the average correlation and p-value.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

corr.series.seg, skel.plot, series.rwl.plot, ccf.series.rwl

Examples

```
library(utils)
data(co021)
dat <- co021
## Create a missing ring by deleting a year of growth in a random series
flagged <- dat$"641143"
flagged <- c(NA, flagged[-325])
names(flagged) <- rownames(dat)
dat$"641143" <- NULL
seg.100 <- corr.series.seg(rwl = dat, series = flagged,
                           seg.length = 100, biweight = FALSE)
## Not run:
flagged2 <- co021$"641143"
names(flagged2) <- rownames(dat)
seg.100.1 <- corr.series.seg(rwl=dat, seg.length=100, biweight=FALSE,
                             series = flagged2)
## Select series by name or column position
seg.100.2 <- corr.series.seg(rwl=co021, seg.length=100, biweight=FALSE,
                             series = "641143")
seg.100.3 <- corr.series.seg(rwl=co021, seg.length=100, biweight=FALSE,
                             series = which(colnames(co021) == "641143"))
identical(seg.100.1, seg.100.2) # TRUE
identical(seg.100.2, seg.100.3) # TRUE

## End(Not run)
```

---

csv2rwl *Read Ring Width File from CSV*

Description

This function reads in a file of ring widths (.rwl) from a text file with comma separated values (csv).

Usage

```
csv2rwl(fname,...)
```

Arguments

fname	a character vector giving the file name of the csv file.
...	other arguments passed to read.table.

Details

This is a simple wrapper to read.table that reads in a text file with ring-width data in "spreadsheet" format. I.e., with series in columns and the the years as rows. The first column should contain the years and each subsequent column should contain a tree-ring series. The series names should be in the first row of the file. The deafult for NA values are empty cells or as the character string "NA" but can also be set using the na.strings argument passed to read.table. E.g.,:

Year	Ser1A	Ser1B	Ser2A	Ser2B
1901	NA	0.45	0.43	0.24
1902	NA	0.05	0.00	0.07
1903	0.17	0.46	0.03	0.21
1904	0.28	0.21	0.54	0.41
1905	0.29	0.85	0.17	0.76
1906	0.56	0.64	0.56	0.31
1907	1.12	1.06	0.99	0.83
etc...				

Note that this is a rudimentary convenience function that isn't doing anything sophisticated. It reads in a file, assigns the years to the row names and sets the class of the object to c("rwl","data.frame") which allows dplR to recognize it.

Although arguments can be passed to read.table, this is not designed to be a flexible function. As is the philosophy with dplR, modifying the code is easy should the user wish to read in a different style of text file (e.g., tab delimited). Typing csv2rwl at the R prompt will get the user started.

Value

Function returns an object of class c("rwl", "data.frame") with the series in columns and the years as rows. The series IDs are the column names and the years are the row names.

Author(s)

Andy Bunn

See Also

read.rwl, read.table

Examples

```
library(utils)
data(ca533)
# write out a rwl file in a format that csv2rwl will understand
tm <- time(ca533)
foo <- data.frame(tm,ca533)
```

```
# this is the temp file where foo will be written
tmpName <- tempfile()
write.csv(foo,file=tmpName,row.names=FALSE)
# read it back in using csv2rwl
bar <- csv2rwl(tmpName)
# check to see if identical
identical(ca533,bar)
# delete temp file
unlink(tmpName)
```

`detrend` *Detrend Multiple Ring-Width Series Simultaneously*

Description

This is a wrapper for `detrend.series` to detrend many ring-width series at once.

Usage

```
detrend(rwl, y.name = names(rwl), make.plot = FALSE,
        method = c("Spline", "ModNegExp", "Mean", "Ar", "Friedman",
                   "ModHugershoff", "AgeDepSpline"),
        nyrs = NULL, f = 0.5, pos.slope = FALSE,
        constrain.nls = c("never", "when.fail", "always"),
        verbose = FALSE, return.info = FALSE,
        wt, span = "cv", bass = 0, difference = FALSE)
```

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`
`y.name`	a `character` vector of `length(ncol(`*rwl*`))` that gives the ID of each series. Defaults to the column names of *rwl*.
`make.plot`	a `logical` flag. Makes plots of the raw data and detrended data if `TRUE`. See details below.
`method`	a `character` vector to determine the detrending methods. See details below. Possible values are all subsets of `c("Spline", "ModNegExp", "Mean", "Ar", "Friedman", "ModHugershoff", "AgeDepSpline")`. Defaults to using all the available methods.
`nyrs`	a number giving the rigidity of the smoothing spline, defaults to 0.67 of series length if *nyrs* is `NULL`.
`f`	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
`pos.slope`	a `logical` flag. Will allow for a positive slope to be used in method `"ModNegExp"` and `"ModHugershoff"`. If `FALSE` the line will be horizontal.

constrain.nls	a character string which controls the constraints of the "ModNegExp" model and the "ModHugershoff". See detrend.series for further details.
verbose	logical. Write out details?
return.info	a logical flag. If TRUE, details about models and data will be added to the return value. See 'Value'.
wt	a numeric vector of case weights for method "Friedman". The default means equals weights.
span	a numeric value controlling method "Friedman", or "cv" (default) for automatic choice by cross-validation. See supsmu.
bass	a numeric value controlling the smoothness of the fitted curve in method "Friedman". See supsmu.
difference	a logical flag. Compute residuals by substraction if TRUE, otherwise use division.

Details

See detrend.series for details on detrending methods. Setting *make.plot* = TRUE will cause plots of each series to be produced. These could be saved using Devices if desired.

Value

If one detrending method is used, a data.frame containing the dimensionless detrended ring widths with column names, row names and dimensions of *rwl*. If more methods are used, a list with ncol(*rwl*) elements each containing a data.frame with the detrended ring widths in each column.

If *return.info* is TRUE, the return value is a list with four parts:

series	the main result described above (data.frame or list of data.frames)
curves	the curve or line used to detrend series. Either a data.frame or a list of data.frames.
model.info	Information about the models corresponding to each output series. A list with one element for each column of *rwl*. See detrend.series ('Value', *model.info*) for a description of the contents.
data.info	Information about the input series. A list with one element for each column of *rwl*. See detrend.series ('Value', *data.info*).

Note

This function uses the foreach looping construct with the %dopar% operator. For parallel computing and a potential speedup, a parallel backend must be registered before running the function. If *verbose* is TRUE, parallel computation is disabled.

Author(s)

Andy Bunn. Improved by Mikko Korpela.

See Also

detrend.series

Examples

```
library(utils)
data(ca533)
## Detrend using modified exponential decay. Returns a data.frame
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
## Detrend using splines and compute
## residuals via subtraction
ca533.rwi <- detrend(rwl = ca533, method = "Spline",
                     difference = TRUE)

## Detrend using modified Hugershoff curve and return info on the model
## fits. Returns a list with: series, curves, modelinfo and data.info
data(co021)
co021.rwi <- detrend(rwl = co021, method = "ModHugershoff",
                     return.info=TRUE)

## Not run:
library(grDevices)
## Detrend using all methods. Returns a list
ca533.rwi <- detrend(rwl = ca533)
## Save a pdf of all series
fname <- tempfile(fileext=".pdf")
print(fname) # tempfile used for output
pdf(fname)
ca533.rwi <- detrend(rwl = ca533, method = c("Spline", "ModNegExp"),
                     make.plot = TRUE)
dev.off()

unlink(fname) # remove the file

## End(Not run)
```

`detrend.series` *Detrend a Ring-Width Series*

Description

Detrend a tree-ring series by one of two methods, a smoothing spline or a statistical model. The series and fits are plotted by default.

Usage

```
detrend.series(y, y.name = "", make.plot = TRUE,
               method = c("Spline", "ModNegExp", "Mean",
               "Ar", "Friedman", "ModHugershoff","AgeDepSpline"),
               nyrs = NULL, f = 0.5, pos.slope = FALSE,
               constrain.nls = c("never", "when.fail", "always"),
               verbose = FALSE, return.info = FALSE,
               wt, span = "cv", bass = 0, difference = FALSE)
```

Arguments

y	a numeric vector. Usually a tree-ring series.
y.name	an optional character vector to name the series for plotting purposes.
make.plot	a logical flag. Makes plots of the raw data and detrended data if TRUE.
method	a character vector to determine the detrending methods. See details below. Possible values are all subsets of c("Spline", "ModNegExp", "Mean", "Ar", "Friedman", "ModHugershoff", "AgeDepSpline"). Defaults to using all the available methods.
nyrs	a number controlling the smoothness of the fitted curve in methods. See Details.
f	a number between 0 and 1 giving the frequency response or wavelength cutoff in method "Spline". Defaults to 0.5. See caps.
pos.slope	a logical flag. Will allow for a positive slope to be used in method "ModNegExp" or "AgeDepSpline". If FALSE the line will be horizontal.
constrain.nls	a character string which controls the constraints of the "ModNegExp" model and and the "ModHugershoff" model which are fit using nonlinear least-squares via nls. The value is an answer to the question: When should the parameters of the nls function be constrained? The options are "never": do not constrain (the default), "when.fail": only compute the constrained solution if the unconstrained fit contains other than positive values, and "always": return the constrained solution, even if the unconstrained one would have been valid. See 'Details'.
verbose	a logical flag. Write out details to the screen?
return.info	a logical flag. If TRUE, details about models and data will be added to the return value. See 'Value'.
wt	a numeric vector of case weights for method "Friedman". The default means equals weights.
span	a numeric value controlling method "Friedman", or "cv" (default) for automatic choice by cross-validation. See supsmu.
bass	a numeric value controlling the smoothness of the fitted curve in method "Friedman". See supsmu.
difference	a logical flag. Compute residuals by subtraction if TRUE, otherwise use division.

Details

This detrends and standardizes a tree-ring series. The detrending is the estimation and removal of the tree's natural biological growth trend. The default standardization is done by dividing each series by the growth trend to produce units in the dimensionless ring-width index (RWI). If difference is TRUE, the index is calculated by subtraction. Values of zero (typically missing rings) in y are set to 0.001.

There are currently seven methods available for detrending although more are certainly possible. The methods implemented are an age-dependent spline via ads (*method* = "AgeDepSpline"), the residuals of an AR model (*method* = "Ar"), Friedman's Super Smoother (*method* = "Friedman"), a simple horizontal line (*method* = "Mean"), or a modified Hugershoff curve (*method* = "ModHugershoff"),

a modified negative exponential curve (*method* = "ModNegExp"), and a smoothing spline via caps (*method* = "Spline").

The "AgeDepSpline" approach uses an age-dependent spline with an initial stiffness of 50 (*nyrs*=50). See ads. If some of the fitted values are not positive then method "Mean" is used. However, this is extremely unlikely.

The "Ar" approach is also known as prewhitening where the detrended series is the residuals of an ar model divided by the mean of those residuals to yield a series with white noise and a mean of one. This method removes all but the high frequency variation in the series and should only be used as such.

The "Friedman" approach uses Friedman's 'super smoother' as implemented in supsmu. The parameters *wt*, *span* and *bass* can be adjusted, but *periodic* is always set to FALSE. If some of the fitted values are not positive then method "Mean" is used.

The "Mean" approach fits a horizontal line using the mean of the series. This method is the fallback solution in cases where the "Spline" or the linear fit (also a fallback solution itself) contains zeros or negative values, which would lead to invalid ring-width indices.

The "ModHugershoff" approach attempts to fit a Hugershoff model of biological growth of the form $f(t) = at^b e^{-gt} + d$, where the argument of the function is time, using nls. See Fritts (2001) for details about the parameters. Option *constrain.nls* gives a possibility to constrain the parameters of the modified negative exponential function. If the constraints are enabled, the nonlinear optimization algorithm is instructed to keep the parameters in the following ranges: $a \geq 0$, $b \geq 0$ and $d \geq 0$. The default is to not constrain the parameters (*constrain.nls* = "never") for nls but warn the user when the parameters go out of range.

If a suitable nonlinear model cannot be fit (function is non-decreasing or some values are not positive) then a linear model is fit. That linear model can have a positive slope unless *pos.slope* is FALSE in which case method "Mean" is used.

The "ModNegExp" approach attempts to fit a classic nonlinear model of biological growth of the form $f(t) = ae^{bt} + k$, where the argument of the function is time, using nls. See Fritts (2001) for details about the parameters. Option *constrain.nls* gives a possibility to constrain the parameters of the modified negative exponential function. If the constraints are enabled, the nonlinear optimization algorithm is instructed to keep the parameters in the following ranges: $a \geq 0$, $b \leq 0$ and $k \geq 0$. The default is to not constrain the parameters (*constrain.nls* = "never") for nls but warn the user when the parameters go out of range.

If a suitable nonlinear model cannot be fit (function is non-decreasing or some values are not positive) then a linear model is fit. That linear model can have a positive slope unless *pos.slope* is FALSE in which case method "Mean" is used.

The "Spline" approach uses a spline where the frequency response is 0.50 at a wavelength of 0.67 * "series length in years", unless specified differently using *nyrs* and *f* in the function caps. If some of the fitted values are not positive then method "Mean" is used.

These methods are chosen because they are commonly used in dendrochronology. There is a rich literature on detrending and many researchers are particularly skeptical of the use of the classic nonlinear model of biological growth ($f(t) = ae^{bt} + k$) for detrending. It is, of course, up to the user to determine the best detrending method for their data.

Note that the user receives a warning if a series has negative values in the fitted curve. This happens fairly commonly with with the 'Ar' method on high-order data. It happens less often with method 'Spline' but isn't unheard of (see 'Examples'). If this happens, users should look carefully at their

data before continuing. Automating detrending and not evaluating each series individually is folly. Remember, frustration over detrending is the number one cause of dendros going to live as hermits in the tallgrass prairie, where there are no trees to worry about.

See the references below for further details on detrending. It's a dark art.

Value

If several methods are used, returns a `data.frame` containing the detrended series (y) according to the methods used. The columns are named and ordered to match *method*. If only one method is selected, returns a vector.

If *return.info* is `TRUE`, the return value is a `list` with four parts:

`series` the main result described above (`data.frame` or vector)

`curves` the curve or line used to detrend `series`. Either a `data.frame` or vector.

`model.info` Information about the models corresponding to each output series. Whereas *series* may return a vector, *model.info* is always a list where each top level element corresponds to one selected method. Also these elements are named and arranged according to the methods selected. Each element is a list with some of the following sub-elements, depending on which detrending method was actually used:

n.zeros See `"data.info"` below. Always present.

zero.years See `"data.info"`. Always present.

method The method actually used for detrending. One of `"Mean"`, `"Line"`, `"ModNegExp"`, `"Spline"` or `"Ar"`. Always present.

mean Mean of the input series, missing values removed. Only for method `"Mean"`.

coefs Coefficients of the model. Methods `"Line"` and `"ModNegExp"`.

formula The `"ModNegExp"` `formula`.

is.constrained A `logical` flag indicating whether the parameters of the `"ModNegExp"` model were constrained. Only interesting when argument *constrain.nls* is set to `"when.fail"`.

nyrs The value of *nyrs* used for `caps`. For methods `"Spline"` and `"AgeDepSpline"`.

order The order of the autoregressive model, selected by AIC (Akaike information criterion). Only for method `"Ar"`.

ar The autoregressive coefficients used by method `"Ar"`. A `numeric` vector ordered by increasing lag.

`data.info` Information about the input series: number (`"n.zeros"`) and location (`"zero.years"`) of zero values. If the locations are in a `character` vector, they are years. Otherwise they are indices to the input series.

`dirtDog` A logical flag indicating whether the requested method resulted in neagtive values for the curve fit, what Cook's ARSTAN called a Dirty Dog.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela. A bug fix related to negative output values is based on work by Alice Cecile.

References

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Fritts, H. C. (2001) *Tree Rings and Climate*. Blackburn. ISBN-13: 978-1-930665-39-2.

See Also

detrend

Examples

```
library(stats)
library(utils)
## Use series CAM011 from the Campito data set
data(ca533)
series <- ca533[, "CAM011"]
names(series) <- rownames(ca533)
# defaults to all six methods
series.rwi <- detrend.series(y = series, y.name = "CAM011", verbose=TRUE)
# see plot with three methods
series.rwi <- detrend.series(y = series, y.name = "CAM011",
                             method=c("Spline", "ModNegExp","Friedman"),
                             difference=TRUE)
# see plot with two methods
# interesting to note difference from ~200 to 250 years
# in terms of what happens to low frequency growth
series.rwi <- detrend.series(y = series, y.name = "CAM011",
                             method=c("Spline", "ModNegExp"))
# see plot with just one method and change the spline
# stiffness to 50 years (which is not neccesarily a good choice!)
series.rwi <- detrend.series(y = series, y.name = "CAM011",
                             method="Spline",nyrs=50)

# note that method "Ar" doesn't get plotted in first panel
# since this approach doesn't approximate a growth curve.
series.rwi <- detrend.series(y = series, y.name = "CAM011",
                             method="Ar")

# note the difference between ModNegExp and ModHugershoff at the
# start of the series. Also notice how curves, etc. are returned
# via return.info
data(co021)
series <- co021[, 4]
names(series) <- rownames(co021)
series.rwi <- detrend.series(y = series, y.name = names(co021)[4],
                             method=c("ModNegExp", "ModHugershoff"),
                             verbose = TRUE, return.info = TRUE,
                             make.plot = TRUE)

# A dirty dog.
# In the case of method=="Spline" the function carries-on
```

```
# and applies method=="Mean" as an alternative.
data(nm046)
series <- nm046[,8]
names(series) <- rownames(nm046)
series.rwi <- detrend.series(y = series, y.name = names(nm046)[8],
                             method="Spline",
                             make.plot = FALSE)
```

`dplR-defunct` *Defunct functions in dplR*

Description

These are functions no longer included in dplR

Details

Function:	plotRings
Function:	ffcsaps, use `caps` instead

`fill.internal.NA` *Fill Internal NA*

Description

This function fills internal `NA` values (i.e., those with numeric data above and below a small data gap) in each column of a `data.frame` such as a data set of ring widths as produced by `read.rwl`.

Usage

```
fill.internal.NA(x, fill = c("Mean", "Spline", "Linear"))
```

Arguments

`x`	a `data.frame` of ring widths with `row.names(x)` containing years and `names(x)` containing each series ID such as produced by `read.rwl`
`fill`	a `numeric` value (e.g., `0`) or a `character` string of `"Mean"`, `"Spline"`, or `"Linear"`. Defaults to `"Mean"`.

Details

There are occasionally data gaps within a tree-ring series. Some of the functions in dplR will fail when an internal NA is encountered (e.g. caps). This function fills internal NA values with either a given numeric value (e.g., 0) or through crude imputation. The latter can be calculated as the mean of the series (fill="Mean") or calculated by fitting a cubic spline (fill="Spline") using the spline function or calculated by linear approximation (fill="Linear") using the function approx.

Editorial: Having internal NA in a tree-ring series is often bad practice and filling those values should be done with caution. For instance, some users code missing rings as NA instead of 0. And missing values (i.e., NA) are sometimes present in maximum latewood density data when the rings are small. A common, but not recommended, practice is to leave stretches of NA values in places where it has been impossible to accurately measure rings (perhaps because of a break in the core). It is often better to treat that core as two separate series (e.g., "01A" and "01B" rather than have internal NA values. As with all processing, the analyst should make a decision based on their experience with the wood and not rely on software to make a choice for them!

Value

A data.frame with colnames(x) and rownames(x). Internal NAs filled as above.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

spline, approx

Examples

```
library(graphics)
library(stats)
foo <- data.frame(x1=c(rnorm(5), NA, NA, rnorm(3)),
                  x2=c(rnorm(10)),
                  x3=c(NA, NA, rnorm(3), NA, rnorm(4)),
                  x4=c(NA, NA, rnorm(3), NA, rnorm(3), NA),
                  x5=c(NA, NA, rnorm(8)),
                  x6=c(NA, rnorm(9)),
                  x7=c(NA, rnorm(5), NA, rnorm(3)),
                  x8=c(rnorm(8), NA, NA),
                  x9=c(rnorm(5), NA, rnorm(3), NA))
row.names(foo) <- 1901:1910
class(foo) <- c("rwl","data.frame")
fill.internal.NA(foo, fill=0)

bar <- fill.internal.NA(foo, fill="Spline")
baz <- fill.internal.NA(foo, fill="Linear")

## note differences in method "Spline" vs. "Linear"
yrs <- time(foo)
```

```
plot(yrs, foo$x7, type="b", lwd=3)
lines(yrs, bar$x7, col="red", lwd=2)
lines(yrs, baz$x7, col="green", lwd=1)
```

`gini.coef` *Calculate the Gini Coefficient*

Description

This function calculates the Gini coefficient on raw or detrended ring-width series.

Usage

```
gini.coef(x)
```

Arguments

`x`	a `numeric` vector

Details

This calculates the Gini coefficient of inequality which is used as an all-lag measure of diversity in tree-ring records – typically detrended series. Lower values indicate lower diversity. The use of the Gini coefficient in dendrochronology is described by Biondi and Qeadan (2008). See Handcock and Morris (1999) for more information.

Value

the Gini coefficient.

Author(s)

Mikko Korpela, based on original by Andy Bunn

References

Biondi, F. and Qeadan, F. (2008) Inequality in Paleorecords. *Ecology*, **89**(4), 1056–1067.

Handcock, M. S. and Morris, M. (1999) *Relative Distribution Methods in the Social Sciences*. Springer. ISBN: 0-387-98778-9.

See Also

`rwl.stats`

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi)
gini.coef(ca533.crn)
```

---

glk *Calculate Gleichläufigkeit*

Description

This function calculates the Gleichläufigkeit and related measures for a given set of tree-ring records.

Usage

```
glk(x, overlap = 50, prob = TRUE)
glk.legacy(x)
```

Arguments

x	a data.frame of tree-ring data with records in columns, and years as rows.
overlap	integer value with minimal length of overlapping growth changes (compared number of tree rings - 1). Comparisons with less overlap are not compared.
prob	if TRUE then the probability of exceedence of the Gleichläufigkeit will be calculated

Details

Gleichläufigkeit is a classical agreement test based on sign tests (Eckstein and Bauch, 1969). This function implements Gleichläufigkeit as the pairwise comparison of all records in data set. This vectorized implementation is faster than the previous version and follows the original definition (Huber 1942), instead of the incorrect interpretation that has been used in the past (Schweingruber 1988, see Buras/Wilmking 2015 for the correction).

The probability of exceedence (p) for the Gleichläufigkeit expresses the chance that the Gleichläufigkeit is incorrect. The observed value of the Gleichläufigkeit is converted to a z-score and based on the standard normal curve the probability of exceedence is calculated. The result is a matrix of all p-values (Jansma 1995, 60-61, see also Visser 2020).

Note that prior to dplR version 1.7.2, glk did not have the overlap or prob and returned a matrix with just the Gleichläufigkeit for all possible combinations of records. That function can still be accessed via glk.legacy.

Value

The funtions returns a named `list` of two or three matrices (p_mat is optional if `prob = TRUE`):

1. glk_mat: `matrix` with Gleichläufigkeit
2. overlap: `matrix` with number of overlapping growth changes.This is the number of overlapping years minus one.
3. p_mat: `matrix` of all probabilities of exceedence for all observed Gleichläufigkeit values.

The matrices can be extracted from the list by selecting the name or index number. If two curves have less than 3 years of overlap, Gleichläufigkeit cannot be computed, and `NA` is returned. To calculate the global glk of the dataset `mean(x$glk_mat, na.rm = TRUE)`.

Author(s)

Christian Zang. Patched and improved by Mikko Korpela. Improved by Allan Buras. Further improved and expanded by Ronald Visser and Andy Bunn

References

Buras, A. and Wilmking, M. (2015) Correcting the calculation of Gleichläufigkeit, *Dendrochronologia* **34**, 29-30. DOI: https://doi.org/10.1016/j.dendro.2015.03.003

Eckstein, D. and Bauch, J. (1969) Beitrag zur Rationalisierung eines dendrochronologischen Verfahrens und zur Analyse seiner Aussagesicherheit. *Forstwissenschaftliches Centralblatt*, **88**(1), 230-250.

Huber, B. (1943) Über die Sicherheit jahrringchronologischer Datierung. *Holz als Roh- und Werkstoff* **6**, 263-268. DOI: https://doi.org/10.1007/BF02603303

Jansma, E., 1995. *RemembeRINGs; The development and application of local and regional tree-ring chronologies of oak for the purposes of archaeological and historical research in the Netherlands*, Nederlandse Archeologische Rapporten 19, Rijksdienst voor het Oudheidkundig Bodemonderzoek, Amersfoort

Schweingruber, F. H. (1988) *Tree rings: basics and applications of dendrochronology*, Kluwer Academic Publishers, Dordrecht, Netherlands, 276 p.

Visser, R.M. (2020) On the similarity of tree-ring patterns: Assessing the influence of semi-synchronous growth changes on the Gleichläufigkeit for big tree-ring data sets,*Archaeometry*, **63**, 204-215 DOI: https://doi.org/10.1111/arcm.12600

See Also

`sgc` sgc is an alternative for `glk`)

Examples

```
library(utils)
  data(ca533)
  ca533.glklist <- glk(ca533)
  ca533.glk_mat <- ca533.glklist$glk_mat
  # calculating the mean GLK including self-similarities
  mean(ca533.glk_mat, na.rm = TRUE)
```

```
# calculating the mean GLK excluding self-similarities
mean(ca533.glk_mat[upper.tri(ca533.glk_mat)], na.rm = TRUE)
```

---

gp.d2pith *Ponderosa Pine Distance to Pith Corresponding to* gp.rwl

---

Description

This data set gives the distance to pith for each series (in mm) that matches the ring widths for gp.rwl – a data set of ponderosa pine (*Pinus ponderosa*) from the Gus Pearson Natural Area (GPNA) in northern Arizona, USA. Data are further described by Biondi and Qeadan (2008) and references therein.

Usage

```
data(gp.d2pith)
```

Format

A data.frame containing series IDs in column 1 (*series*) and the distance (in mm) from the innermost ring to the pith of the tree (*d2pith*). This can be used together with the ring widths to calculate the area of each ring.

Source

DendroLab, University of Nevada Reno, USA. https://dendrolaborg.wordpress.com/

References

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

---

gp.dbh *Ponderosa Pine Stem Diameters and Bark Thickness (*gp.rwl*)*

Description

This data set gives the diameter at breast height for each series that matches the series in gp.rwl – a data set of ponderosa pine (*Pinus ponderosa*) from the Gus Pearson Natural Area (GPNA) in northern Arizona, USA. Data are further described by Biondi and Qeadan (2008) and references therein.

Usage

```
data(gp.dbh)
```

Format

A `data.frame` containing series IDs in column 1 (*series*), tree diameter (in cm) at breast height (*dbh*), and the bark thickness (in cm). This can be used together with the ring widths to calculate the area of each ring.

Source

DendroLab, University of Nevada Reno, USA. https://dendrolaborg.wordpress.com/

References

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

`gp.po` *Ponderosa Pine Pith Offsets Corresponding to* `gp.rwl`

Description

This data set gives the pith offsets that match the ring widths for `gp.rwl` – a data set of ponderosa pine (*Pinus ponderosa*) from the Gus Pearson Natural Area (GPNA) in northern Arizona, USA. Data are further described by Biondi and Qeadan (2008) and references therein.

Usage

```
data(gp.po)
```

Format

A `data.frame` containing series IDs in column 1 (*series*) and the number of years between the beginning of that series in `gp.rwl` and the pith of the tree (*pith.offset*). This can be used together with the ring widths to calculate the cambial age of each ring.

Source

DendroLab, University of Nevada Reno, USA. https://dendrolaborg.wordpress.com/

References

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

gp.rwl *Ponderosa Pine Ring Widths from Gus Pearson Natural Area*

Description

This data set includes ring-width measurements for ponderosa pine (*Pinus ponderosa*) increment cores collected at the Gus Pearson Natural Area (GPNA) in northern Arizona, USA. There are 58 series from 29 trees (2 cores per tree). Data are further described by Biondi and Qeadan (2008) and references therein.

Usage

```
data(gp.rwl)
```

Format

A `data.frame` containing 58 ring-width series in columns and 421 years in rows.

Source

DendroLab, University of Nevada Reno, USA. https://dendrolaborg.wordpress.com/

References

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

hanning *Hanning Filter*

Description

Applies a Hanning filter of length n to x.

Usage

```
hanning(x, n = 7)
```

Arguments

x	a vector
n	length of the Hanning filter, defaults to 7

Details

This applies a low frequency Hanning (a.k.a. Hann) filter to x with weight set to n.

Value

A filtered vector.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Oppenheim, A. V., Schafer, R. W., and Buck, J. R. (1999) *Discrete-Time Signal Processing*. Prentice-Hall, 2nd edition. ISBN-13: 978-0-13-754920-7.

See Also

`filter`

Examples

```
library(graphics)
library(utils)
data(ca533)
yrs <- time(ca533)
y <- ca533[, 1]
not.na <- !is.na(y)
yrs <- yrs[not.na]
y <- y[not.na]
plot(yrs, y, xlab = "Years", ylab = "Series1 (mm)",
     type = "l", col = "grey")
lines(yrs, hanning(y, n = 9), col = "red", lwd = 2)
lines(yrs, hanning(y, n = 21), col = "blue", lwd = 2)
legend("topright", c("Series", "n=9", "n=21"),
       fill=c("grey", "red", "blue"))
```

---

i.detrend *Interactively Detrend Multiple Ring-Width Series*

Description

Interactively detrend multiple tree-ring series by one of two methods, a smoothing spline or a statistical model. This is a wrapper for `detrend.series`.

Usage

```
i.detrend(rwl, y.name = names(rwl), nyrs = NULL, f = 0.5,
          pos.slope = FALSE)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl or ca533
y.name	a character vector of length ncol(*rwl*) that gives the ID of each series. Defaults to the column names of *rwl*.
nyrs	a number giving the rigidity of the smoothing spline, defaults to 0.67 of series length if *nyrs* is NULL.
f	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
pos.slope	a logical flag. Will allow for a positive slope to be used in method "ModNegExp". If FALSE the line will be horizontal.

Details

This function allows a user to choose detrending curves based on plots that are produced by detrend.series for which it is essentially a wrapper. The user enters their choice of detrended method via keyboard at a prompt for each ring width series in *rwl*. See detrend.series for examples and details on the detrending methods.

Value

A data.frame containing each detrended series according to the method used as columns and rownames set to colnames(*y*). These are typically years. Plots are also produced as the user chooses the detrending methods through keyboard input.

Author(s)

Andy Bunn

See Also

detrend.series

---

i.detrend.series *Interactively Detrend a Ring-Width Series*

Description

Interactively detrend a tree-ring series by one of three methods, a smoothing spline, a linear model, or the mean. This is a wrapper for detrend.series.

Usage

```
i.detrend.series(y, y.name = NULL, nyrs = NULL, f = 0.5,
                 pos.slope = FALSE)
```

Arguments

y	a numeric vector. Usually a tree-ring series.
y.name	an optional character vector to name the series for plotting purposes.
nyrs	a number giving the rigidity of the smoothing spline, defaults to 0.67 of series length if *nyrs* is NULL.
f	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
pos.slope	a logical flag. Will allow for a positive slope to be used in method "ModNegExp". If FALSE the line will be horizontal.

Details

This function allows a user to choose a detrending method based on a plot that is produced by detrend.series for which it is essentially a wrapper. The user enters their choice of detrended method via keyboard at a prompt. See detrend.series for examples and details on the detrending methods.

Value

A vector containing the detrended series (y) according to the method used with names set to colnames(y). These are typically years. A plot is also produced and the user chooses a method through keyboard input.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

detrend.series

---

insert.ring *Edit a Ring-Width Series*

Description

Insert or delete rings from a ring-width series

Usage

```
insert.ring(rw.vec,rw.vec.yrs=as.numeric(names(rw.vec)),
            year,ring.value=mean(rw.vec,na.rm=TRUE),
            fix.last=TRUE,fix.length=TRUE)
delete.ring(rw.vec,rw.vec.yrs=as.numeric(names(rw.vec)),
            year,fix.last=TRUE,fix.length=TRUE)
```

Arguments

rw.vec	a vector of data
rw.vec.yrs	the years for rw.vec as names
year	the year to add or delete
ring.value	the value to add
fix.last	logical. If TRUE the last year of the series is fixed and the first year changes.
fix.length	logical. If TRUE the length of the output will be the length of the input.

Details

Simple editing of ring widths.

Value

A named vector.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

dplR

Examples

```
library(utils)
data(gp.rwl)
dat <- gp.rwl
# insert a value of zero for the year 1950 in series 50A
# fix the last year of growth and maintain the length of the series
tmp <- insert.ring(rw.vec=dat$"50A",rw.vec.yrs = time(dat),
                   year=1950,ring.value=0,fix.length = TRUE)
# with fix.length=TRUE this can be merged back into the rwl object:
data.frame(dat$"50A",tmp)
dat$"50A" <- tmp

# note that if fix.last = FALSE and fix.length = FALSE inserting a ring causes the
# ending year of the series to be pushed forward and the length of the output to
# be longer than the original series.
tmp <- insert.ring(rw.vec=dat$"50A",rw.vec.yrs = time(dat),
                   year=1950,ring.value=0, fix.last = FALSE,
                   fix.length = FALSE)
# with fix.length=FALSE this can't be merged back into the rwl object the
# same way as above
tail(tmp)
length(tmp)
nrow(dat)
```

```
# the same logic applies to deleting rings.
dat <- gp.rwl
# delete the year 1950 in series 50A
# fix the last year of growth and maintain the length of the series
tmp <- delete.ring(rw.vec=dat$"50A",rw.vec.yrs = time(dat),
                   year=1950,fix.last = TRUE, fix.length = TRUE)
# with fix.length=TRUE this can be merged back into the rwl object:
data.frame(dat$"50A",tmp)
dat$"50A" <- tmp

# note that if fix.last = FALSE and fix.length = FALSE inserting a ring causes the
# ending year of the series to be pushed forward and the length of the output to
# be longer than the original series.
tmp <- delete.ring(rw.vec=dat$"50A", rw.vec.yrs = time(dat),
                   year=1950, fix.last = FALSE,
                   fix.length = FALSE)
# with fix.length=FALSE this can't be merged back into the rwl object the
# same way as above
tail(tmp)
length(tmp)
nrow(dat)
```

`interseries.cor` *Individual Series Correlation Against a Master Chronology*

Description

This function calculates the correlation between a series and a master chronology.

Usage

```
interseries.cor(rwl,n=NULL,prewhiten=TRUE,biweight=TRUE,
  method = c("spearman", "pearson", "kendall"))
```

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`.
`n`	`NULL` or an integral value giving the filter length for the `hanning` filter used for removal of low frequency variation.
`prewhiten`	`logical` flag. If `TRUE` each series is whitened using `ar`.
`biweight`	`logical` flag. If `TRUE` then a robust mean is calculated using `tbrm`.
`method`	Can be either `"pearson"`, `"kendall"`, or `"spearman"` which indicates the correlation coefficient to be used. Defaults to `"spearman"`. See `cor.test`.

Details

This function calculates correlation serially between each tree-ring series and a master chronology built from all the other series in the *rwl* object (leave-one-out principle).

Each series in the rwl object is optionally detrended as the residuals from a `hanning` filter with weight *n*. The filter is not applied if *n* is `NULL`. Detrending can also be done via prewhitening where the residuals of an `ar` model are added to each series mean. This is the default. The master chronology is computed as the mean of the *rwl* object using `tbrm` if *biweight* is `TRUE` and `rowMeans` if not. Note that detrending can change the length of the series. E.g., a `hanning` filter will shorten the series on either end by `floor(n/2)`. The prewhitening default will change the series length based on the `ar` model fit. The effects of detrending can be seen with `series.rwl.plot`.

This function produces the same output of the *overall* portion of `corr.rwl.seg`. The mean correlation value given is sometimes referred to as the "overall interseries correlation" or the "COFECHA interseries correlation". This output differs from the `rbar` statistics given by `rwi.stats` in that `rbar` is the average pairwise correlation between series where this is the correlation between a series and a master chronology.

Value

a `data.frame` with correlation values and p-values given from `cor.test`

Author(s)

Andy Bunn, patched and improved by Mikko Korpela

See Also

`rwl.stats`, `rwi.stats`

Examples

```
library(utils)
data(gp.rwl)
foo <- interseries.cor(gp.rwl)
# compare to:
# corr.rwl.seg(rwl=gp.rwl,make.plot=FALSE)$overall
# using pearson's r
foo <- interseries.cor(gp.rwl,method="pearson")

# two measures of interseries correlation
# compare interseries.cor to rbar from rwi.stats
gp.ids <- read.ids(gp.rwl, stc = c(0, 2, 1))
bar <- rwi.stats(gp.rwl, gp.ids, prewhiten=TRUE)
bar$rbar.eff
mean(foo[,1])
```

latexDate *Date Conversion to Character in LaTeX Format*

Description

This is a simple convenience function that returns a date in the format used by '\today' in LaTeX. A possible use case is fixing the date shown in a vignette at weaving time.

Usage

```
latexDate(x = Sys.Date(), ...)
```

Arguments

x	any object for which an as.POSIXlt method exists. Defaults to the current date.
...	other arguments to as.POSIXlt

Value

A character vector

Author(s)

Mikko Korpela

Examples

```
latexDate()                              # today
latexDate(Sys.Date() + 5)                # today + 5 days
latexDate(c("2013-12-06", "2014-09-19")) # fixed dates
## [1] "December 6, 2013"   "September 19, 2014"
latexDate(5*60*60*24, origin=Sys.Date()) # today + 5 days
```

latexify *Convert Character Strings for Use with LaTeX*

Description

Some characters cannot be entered directly into a LaTeX document. This function converts the input character vector to a form suitable for inclusion in a LaTeX document in text mode. It can be used together with '\Sexpr' in vignettes.

Usage

```
latexify(x, doublebackslash = TRUE, dashdash = TRUE,
         quotes = c("straight", "curved"),
         packages = c("fontenc", "textcomp"))
```

Arguments

x	a character vector
doublebackslash	a logical flag. If TRUE, backslashes in the output are doubled. It seems that Sweave needs TRUE and knitr FALSE.
dashdash	a logical flag. If TRUE (the default), consecutive dashes ("-") in the input will be rendered as separate characters. If FALSE, they will not be given any special treatment, which will usually mean that two dashes are rendered as an en dash and three dashes make an em dash.
quotes	a character string specifying how single and double quotes (ASCII codes 39 and 34) and stand-alone grave accents (ASCII code 96) in the input are treated. The default is to use straight quotes and the proper symbol for the grave accent. The other option is to use curved right side (closing) quotes, and let LaTeX convert the grave accent to opening curved quotes. Straight double quotes are not available in the default OT1 font encoding of LaTeX. Straight single quotes and the grave accent symbol require the "textcomp" package. See *packages*.
packages	a character vector specifying the LaTeX packages allowed. The use of some symbols in LaTeX requires commands or characters made available by an add-on package. If a package required for a given character is not marked as available, a fallback solution is silently used. For example, curved quotes may be used instead of straight quotes. The supported packages are "eurosym" (not used by default), "fontenc" and "textcomp". Including "fontenc" in the vector means that some other encoding than OT1 is going to be used. Note that straight quotes are an exception in the sense that a reasonable substitute (curved quotes) is available. In many other cases, "textcomp" and "fontenc" are silently assumed.

Details

The function is intended for use with unformatted inline text. Newlines, tabs and other whitespace characters ("[:space:]" in regex) are converted to spaces. Control characters ("[:cntrl:]") that are not whitespace are removed. Other more or less special characters in the ASCII set are '{', '}', '\', '#', '$', '%', '^', '&', '_', '~', double quote, '/', single quote, '<', '>', '|', grave and '-'. They are converted to the corresponding LaTeX commands. Some of the conversions are affected by user options, e.g. *dashdash*.

Before applying the substitutions described above, input elements with Encoding set to "bytes" are printed and the output is stored using captureOutput. The result of this intermediate stage is ASCII text where some characters are shown as their byte codes using a hexadecimal pair prefixed with "\x". This set includes tabs, newlines and control characters. The substitutions are then applied to the intermediate result.

The quoting functions sQuote and dQuote may use non-ASCII quote characters, depending on the locale. Also these quotes are converted to LaTeX commands. This means that the quoting functions are safe to use with any LaTeX input encoding. Similarly, some other non-ASCII characters, e.g. letters, currency symbols, punctuation marks and diacritics, are converted to commands.

Adding "eurosym" to *packages* enables the use of the euro sign as provided by the "eurosym" package ('\euro').

The result is converted to UTF-8 encoding, Normalization Form C (NFC). Note that this function will not add any non-ASCII characters that were not already present in the input. On the contrary, some non-ASCII characters, e.g. all characters in the "latin1" (ISO-8859-1) Encoding (character set), are removed when converted to LaTeX commands. Any remaining non-ASCII character has a good chance of working when the document is processed with XeTeX or LuaTeX, but the Unicode support available with pdfTeX is limited.

Assuming that 'pdflatex' is used for compilation, suggested package loading commands in the document preamble are:

```
\usepackage[T1]{fontenc}    % no '"' in OT1 font encoding
\usepackage{textcomp}       % some symbols e.g. straight single quote
\usepackage[utf8]{inputenx} % UTF-8 input encoding
\input{ix-utf8enc.dfu}      % more supported characters
```

Value

A character vector

Author(s)

Mikko Korpela

References

INRIA. Tralics: a LaTeX to XML translator, HTML documentation of all TeX commands. https://www-sop.inria.fr/marelle/tralics/.

Levitt, N., Persch, C., and Unicode, Inc. (2013) GNOME Character Map, application version 3.10.1.

Mittelbach, F., Goossens, M., Braams, J., Carlisle, D., and Rowley, C. (2004) *The LaTeX Companion*. Addison-Wesley, second edition. ISBN-13: 978-0-201-36299-2.

Pakin, S. (2009) The Comprehensive LaTeX Symbol List. https://www.ctan.org/tex-archive/info/symbols/comprehensive.

The Unicode Consortium. The Unicode Standard. https://home.unicode.org/.

Examples

```
x1 <- "clich\xe9\nma\xf1ana"
Encoding(x1) <- "latin1"
x1
x2 <- x1
Encoding(x2) <- "bytes"
x2
x3 <- enc2utf8(x1)
testStrings <-
    c("different     kinds\nof\tspace",
      "control\a characters \ftoo",
      "{braces} and \\backslash",
      '#various$ %other^ &characters_ ~escaped"/coded',
      x1,
```

```
      x2,
      x3)
latexStrings <- latexify(testStrings, doublebackslash = FALSE)
## All should be "unknown"
Encoding(latexStrings)
cat(latexStrings, sep="\n")
## Input encoding does not matter
identical(latexStrings[5], latexStrings[7])
```

`morlet` *Perform a Continuous Morlet Wavelet Transform*

Description

This function performs a continuous wavelet transform on a time series.

Usage

```
morlet(y1, x1 = seq_along(y1), p2 = NULL, dj = 0.25, siglvl = 0.95)
```

Arguments

`y1`	`numeric` vector. Series to be transformed.
`x1`	`numeric`. A vector of values giving the years for the plot. Must be the same length as `length(`*y1*`)`.
`p2`	`numeric`. The number of power of two to be computed for the wavelet transform. Calculated from length of *y1* if `NULL`.
`dj`	`numeric`. Sub-octaves per octave calculated.
`siglvl`	`numeric`. Level for the significance test.

Details

This performs a continuous wavelet transform of a time series. This function is typically invoked with `wavelet.plot`.

Value

A `list` containing:

`y`	`numeric`. The original time series.
`x`	`numeric`. The time values.
`wave`	`complex`. The wavelet transform.
`coi`	`numeric`. The cone of influence.
`period`	`numeric`. The period.
`Scale`	`numeric`. The scale.
`Signif`	`numeric`. The significant values.
`Power`	`numeric`. The squared power.

Note

This is a port of Torrence's IDL code, which can be accessed through the Internet Archive Wayback Machine.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Torrence, C. and Compo, G. P. (1998) A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, **79**(1), 61–78.

See Also

`wavelet.plot`

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi, prewhiten = FALSE)
Years <- time(ca533.crn)
CAMstd <- ca533.crn[, 1]
out.wave <- morlet(y1 = CAMstd, x1 = Years, dj = 0.1, siglvl = 0.99)
```

---

net *Calculate NET*

Description

Computes the *NET* parameter for a set of tree-ring records or other time-series data.

Usage

```
net(x, weights = c(v = 1, g = 1))
```

Arguments

`x`	A `matrix` or `data.frame` with at least two rows and two columns containing `numeric` data. The rows should represent a sequence of sampling points with uniform intervals (e.g. a range of years), but this is not checked. Each column is a time-series spanning either the whole time range or a part of it.
`weights`	A `numeric` vector with two elements. Normally, variation (`"v"`) and Gegenläufigkeit (`"g"`) contribute to NET with equal weight. It is possible to use different weights by setting them here. The names of the vector are matched to `c("v", "g")` (see 'Examples'). If no names are given, the first element is the weight of variation.

Details

This function computes the NET parameter (Esper et al., 2001). The overall NET is an average of all (non-NA) yearly values NET_j, which are computed as follows:

$$NET_j = v_j + (1 - G_j)$$

The yearly variation v_j is the standard deviation of the measurements of a single year divided by their mean. Gegenläufigkeit $1 - G_j$ is based on one definition of Gleichläufigkeit G_j, similar to but not the same as what glk computes. Particularly, in the formula used by this function (Esper et al., 2001), simultaneous zero differences in two series are not counted as a synchronous change.

The weights of v_j and $1 - G_j$ in the sum can be adjusted with the argument *weights* (see above). As a rather extreme example, it is possible to isolate variation or Gegenläufigkeit by setting one of the weights to zero (see 'Examples').

Value

A list with the following components, in the same order as described here:

all	a numeric vector containing NET_j. Row names of *x* (if any) are copied here.
average	a numeric value NET, the average of the "all" vector (NA values removed).

Author(s)

Mikko Korpela

References

Esper, J., Neuwirth, B., and Treydte, K. (2001) A new parameter to evaluate temporal signal strength of tree-ring chronologies. *Dendrochronologia*, **19**(1), 93–102.

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.net <- net(ca533.rwi)
tail(ca533.net$all)
ca533.net$average
## Not run:
## Isolate the components of NET
ca533.v <- net(ca533.rwi, weights=c(v=1,0))
ca533.g <- net(ca533.rwi, weights=c(g=1,0))

## End(Not run)
```

nm046 *Los Alamos Tree Ring Widths*

Description

This data set gives the raw ring widths for Douglas fir *Pseudotsuga menziesii* near Los Alamos New Mexico, USA. There are 8 series. Data set was created using `read.rwl` and saved to an .rda file using `save`.

Usage

```
data(nm046)
```

Format

A `data.frame` containing 8 tree-ring series in columns and 289 years in rows.

Source

International tree-ring data bank, Accessed on 20-April-2021 at https://www.ncei.noaa.gov/pub/data/paleo/treering/measurements/northamerica/usa/nm046.rwl

References

O'Brien, D (2002) Los Alamos Data Set. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series 2002-NM046.RWL. NOAA/NCDC Paleoclimatology Program, Boulder, Colorado, USA.

pass.filt *Low-pass, high-pass, band-pass, and stop-pass filtering*

Description

Applies low-pass, high-pass, band-pass, or stop-pass filtering to y with frequencies (or periods) supplied by the user.

Usage

```
pass.filt(y, W, type = c("low", "high", "stop", "pass"),
          method = c("Butterworth", "ChebyshevI"),
          n = 4, Rp = 1)
```

Arguments

y	a numeric vector, typically a tree-ring series.
W	a numeric vector giving frequency or period of the filter. See details.
type	a character giving the type of filter. Values can be "low", "high", "stop", or "pass" for low-pass, high-pass, band-pass, or stop-pass filters. Defaults to "low".
method	a character specifying indicating whether to use a Butterworth (default) or a type I Chebyshev filter.
n	a numeric value giving the order of the filter. Larger numbers create steeper fall off.
Rp	a numeric value giving the dB for the passband ripple.

Details

This function applies either a Butterworth or a Chebyshev type I filter of order *n* to a signal and is nothing more than a wrapper for functions in the signal package. The filters are designed via butter and cheby1. The filter is applied via filtfilt.

The input data (y) has the mean value subtracted and is then padded via reflection at the start and the end to a distance of twice the maximum period. The padded data and the filter are passed to filtfilt after which the data are unpadded and returned afer the mean is added back.

The argumement *W* can be given in either frequency between 0 and 0.5 or, for convenience, period (minimum value of 2). For low-pass and high-pass filters, *W* must have a length of one. For low-pass and high-pass filters *W* must be a two-element vector (c(low, high)) specifying the lower and upper boundaries of the filter.

Because this is just a wrapper for casual use with tree-ring data the frequencies and periods assume a sampling frequency of one. Users are encouraged to build their own filters using the signal package.

Value

A filtered vector.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

hanning, detrend

Examples

```
data("co021")
x <- na.omit(co021[, 1])

# 20-year low-pass filter -- note freq is passed in
bSm <- pass.filt(x, W=0.05, type="low", method="Butterworth")
cSm <- pass.filt(x, W=0.05, type="low", method="ChebyshevI")
```

```
plot(x, type="l", col="grey")
lines(bSm, col="red")
lines(cSm, col="blue")

# 20-year high-pass filter -- note period is passed in
bSm <- pass.filt(x, W=20, type="high")
plot(x, type="l", col="grey")
lines(bSm, col="red")

# 20 to 100-year band-pass filter -- note freqs are passed in
bSm <- pass.filt(x, W=c(0.01, 0.05), type="pass")
cSm <- pass.filt(x, W=c(0.01, 0.05), type="pass", method="ChebyshevI")
plot(x, type="l", col="grey")
lines(bSm, col="red")
lines(cSm, col="blue")

# 20 to 100-year stop-pass filter -- note periods are passed in
cSm <- pass.filt(x, W=c(20, 100), type="stop", method="ChebyshevI")
plot(x, type="l", col="grey")
lines(cSm, col="red")
```

`plot.crn` *Plot a Tree-Ring Chronology*

Description

This function makes a default plot of a tree-ring chronology from a `data.frame` of the type produced by `chron`, `chron.ars`, `chron.stabilized`, `ssf`.

Usage

```
## S3 method for class 'crn'
plot(x, add.spline = FALSE, nyrs = NULL, ...)
```

Arguments

`x`	a `data.frame` of `class{crn}`.e.g., as produced by `chron`, `chron.ars`, `chron.stabilized`, `ssf`.
`add.spline`	a `logical` flag. Will add a line with a smoothing spline using `caps`
`nyrs`	a number giving the rigidity of the smoothing spline. Defaults to 1/3 times the length of the first chronology if *nyrs* is `NULL`
`...`	Additional arguments to pass to `plot`.

Details

This makes a crude plot of one or more tree-ring chronologies.

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

chron

Examples

```
library(graphics)
data(wa082)
# Truncate the RW data to a sample depth at least 5
wa082Trunc <- wa082[rowSums(!is.na(wa082))>4,]
# Detrend with age-dependent spline
wa082RWI <- detrend(wa082Trunc,method = "AgeDep")
# make several chronologies
wa082CRN1 <- chron(wa082RWI)
wa082CRN2 <- chron.stabilized(wa082RWI,
                              winLength=51,
                              biweight = TRUE,
                              running.rbar = TRUE)
wa082CRN3 <- chron.ars(wa082RWI)
wa082CRN4 <- ssf(wa082Trunc)

# and plot
plot.crn(wa082CRN1,add.spline = TRUE,nyrs=20)
plot.crn(wa082CRN2,add.spline = TRUE,nyrs=20)
plot(wa082CRN3,add.spline = TRUE,nyrs=20)
plot(wa082CRN4,add.spline = TRUE,nyrs=20)

# a custom crn
foo <- data.frame(wa082CRN1,sfc=wa082CRN4$sfc)
foo <- foo[,c(1,3,2)]
class(foo) <- c("crn","data.frame")
plot.crn(foo,add.spline = TRUE,nyrs=20)
```

---

plot.crs *Plotting crs Objects*

Description

Plots objects returned from corr.rwl.seg.

Usage

```
## S3 method for class 'crs'
plot(x, ...)
```

Arguments

x	An object of class "crs".
...	Additional arguments passed to plot

Value

None. A plot is produced.

Author(s)

Andy Bunn

See Also

corr.rwl.seg

Examples

```
library(graphics)
data(co021)
foo <- corr.rwl.seg(co021, make.plot = FALSE)
plot(foo)
```

---

plot.rwl *Plotting Rwl Objects*

Description

Plots rwl objects.

Usage

```
## S3 method for class 'rwl'
plot(x, plot.type=c("seg","spag"), ...)
```

Arguments

x	An object of class "rwl".
plot.type	Character. Type "seg" calls seg.plot while "spag" calls spag.plot
...	Additional arguments for each type

Value

None. A plot is produced.

Author(s)

Andy Bunn

See Also

read.rwl

Examples

```
library(graphics)
library(utils)
data(co021)
plot(co021, plot.type="seg")
plot(co021, plot.type="spag")
plot(co021, plot.type="spag", zfac=2)
```

---

po.to.wc *Convert Pith Offset to Wood Completeness*

Description

This function creates a partial wood completeness data structure based on pith offset data.

Usage

```
po.to.wc(po)
```

Arguments

po	A data.frame containing two variables. Variable one (*series*) gives the series ID as either characters or factors. Variable two (*pith.offset*) contains integral values and gives the years from the beginning of the core to the pith (or center) of the tree. The minimum value is 1.

Details

Uses *pith.offset* - 1 as the number of missing heartwood rings.

Value

A data.frame containing one variable of wood completeness data: *n.missing.heartwood* (integer type). This can be used as input to write.tridas.

Author(s)

Mikko Korpela

See Also

wc.to.po, rcs, write.tridas

Examples

```
## Not run:
library(utils)
data(gp.po)
all(wc.to.po(po.to.wc(gp.po)) == gp.po)

## End(Not run)
```

---

pointer *Calculates Pointer Years from a Group of Ring-Width Series*

Description

This function calculates pointer years on a data.frame of ring-width series using the Becker algorithm. The pointer years are computed with adjustable thresholds of relative radial growth variation and number of series displaying similar growth pattern (i.e. positive or negative variations).

Usage

```
pointer(rwl, rgv.thresh = 10, nseries.thresh = 75, round.decimals = 2)
```

Arguments

rwl	a data.frame with ring-width series as columns and years as rows such as that produced by read.rwl.
rgv.thresh	a numeric giving the minimum absolute relative radial growth variation (in percentage) above which the growth change from the year *t-1* to *t* is considered as significant. Must be > 0. Values > 100 are possible but highly unusual. See references. Defaults to 10.
nseries.thresh	a numeric giving the minimum percentage of series displaying significant relative radial growth variations of a same sign above which the year *t* is considered as a pointer year. Positive significant variations will results in a positive pointer year, negative ones in a negative pointer year. This number ranges from 1 to 100. Defaults to 75.
round.decimals	an integer indicating the number of decimal places to be used for outputs. This number must be positive. Defaults to 2.

Details

This calculates pointer years from ring-width series for each year *t* of the time period covered by the series using the Becker algorithm. This algorithm is based on, first, the calculation of the individual relative radial growth variation by comparison of ring-width of year *t* to that of year *t*-1 for each series, and second, the inter-series comparison of both sign and magnitude of these variations.

For example, if *rgv.thresh* and *nseries.thresh* are set at 10 and 75 respectively, pointer years will be defined as those years when at least 75% of the series present an absolute relative radial growth variation higher than 10%.

Users unfamiliar with the Becker algorithm should refer to Becker et al. (1994) and Mérian and Lebourgeois (2011) for further details.

Value

A `data.frame` containing the following columns (each row corresponds to one position of the window):

`Year`	Considered year (t).
`Nb.series`	Number of available series.
`Perc.pos`	Percentage of series displaying a significant positive radial growth variation.
`Perc.neg`	Percentage of series displaying a significant negative radial growth variation.
`Nature`	Number indicating whether the year is a positive pointer year (1), a negative pointer year (-1) or a regular year (0).
`RGV_mean`	Mean radial growth variations over the available series.
`RGV_sd`	Standard deviation of the radial growth variations over the available series.

Author(s)

Pierre Mérian. Improved by Mikko Korpela and Andy Bunn.

References

Becker, M., Nieminen, T. M., and Gérémia, F. (1994) Short-term variations and long-term changes in oak productivity in northeastern France – the role of climate and atmospheric CO2. *Annals of Forest Science*, **51**(5), 477–492.

Mérian, P. and Lebourgeois, F. (2011) Size-mediated climate–growth relationships in temperate forests: A multi-species analysis. *Forest Ecology and Management*, **261**(8), 1382–1391.

See Also

`skel.plot`

Examples

```
## Pointer years calculation on ring-width series. Returns a data.frame.
library(utils)
data(gp.rwl)
py <- pointer(rwl=gp.rwl, rgv.thresh=10, nseries.thresh=75,
              round.decimals=2)
tail(py)
```

---

powt *Power Transformation of Tree-Ring Data*

Description

Power transformation of tree-ring width.

Usage

```
powt(rwl, method = "universal", rescale = FALSE,
     return.power=FALSE)
```

Arguments

rwl	a data.frame, typically of raw tree-ring widths series, such as that produced by read.rwl or read.fh
method	a character vector to determine the transform method. See details below. Possible values are either "universal" or "cook." Pattern matching is used.
rescale	logical flag. If TRUE then each transformed series is rescaled to have the orginal mean and standard deviation of the input data. See details.
return.power	logical flag. If TRUE then the power estimate(s) is returned.

Details

In dendrochronology, ring width series are sometimes power transformed to address heteroscedasticity.

The classic procedure used by method="cook" is a variance stabilization technique implemented after Cook & Peters (1997): for each series a linear model is fitted on the logs of level and spread, where level is defined as the local mean $M_t = (R_t + R_{t-1})/2$ with ring widths R, and spread S is the local standard deviation defined as $S_t = |R_t - R_{t-1}|$. The regression coefficient b from a linear model $\log S = k + b \log M$ is then used for the power transform $\star R_t = R_t^{1-b}$.

The procedure above is modified with method="universal" where all samples are used simultaneously in a linear mixed-effects model with time (year) as a random effect: lmer(log S ~ log M + (1|year). This "universal" or "signal free" approach accounts for the common year effect across all of the series in rwl and should address that not every year has the same change in environmental conditions to the previous year.

The rescale argument will return the series with a mean and standard deviation that matches the input data. While this is a common convention, users should note that this can produce negative values which can be confusing if thought of as "ring widths."

Value

Either an object of class `c("rwl", "data.frame")` containing the power transformed ring width series with the series in columns and the years as rows or in the case of a single series, a possibly named vector of the same. With class `rwl`, the series IDs are the column names and the years are the row names.

If `return.power=TRUE` the returned object is a `list` containing the power transformed data and a `numeric` with the power estimate(s) used to transform the data.

Author(s)

Christian Zang implemented the Cook and Peters method. Stefan Klesse conceived and wrote the universal method. Patched and improved by Mikko Korpela and Andy Bunn.

References

Cook, E. R. and Peters, K. (1997) Calculating unbiased tree-ring indices for the study of climatic and environmental change. *The Holocene*, **7**(3), 361–370.

Examples

```
library(utils)
data(zof.rwl)
powtUniversal <- powt(zof.rwl, method = "universal")
powtCook <- powt(zof.rwl, method = "cook")

op <- par(no.readonly = TRUE)
par(mfcol = c(1, 3))
hist(summary(zof.rwl)$skew,
     breaks = seq(-2.25,2.25,by=0.25),
     main="Raw Data",xlab="Skew")
hist(summary(powtUniversal)$skew,
     breaks = seq(-2.25,2.25,by=0.25),
     main="Universal POWT",xlab="Skew")
hist(summary(powtCook)$skew,
     breaks = seq(-2.25,2.25,by=0.25),
     main="Cook POWT",xlab="Skew")
par(op) # restore graphical parameters
```

---

print.redfit *Printing Redfit Results*

Description

Print information contained in or derived from a redfit object.

Usage

```
## S3 method for class 'redfit'
print(x, digits = NULL, csv.out = FALSE, do.table = FALSE,
      prefix = "", row.names = FALSE, file = "", ...)
```

Arguments

x	An object of class "redfit".
digits	Specifies the desired number of significant digits in the output. The argument is passed to format and print.data.frame. A positive integral value or NULL. If NULL, the value in options("digits") is used.
csv.out	A logical flag. If TRUE, writes a large, comma-separated table of information. The table contains one row for each frequency. If FALSE, writes a few summary numbers instead. See 'Details'.
do.table	A logical flag. If TRUE, the large information table is also printed when csv.out is FALSE, although not in a comma-separated format but with print.data.frame.
prefix	A prefix to be used on every output line except the large information table. REDFIT (see References) uses "# ".
row.names	A logical flag enabling or disabling automatic row names from the large information table, or a character vector of names. In any case, the table will contain frequency as the first column.
file	A writable connection or a character string naming a file. Used for setting the output destination when *csv.out* is TRUE. The default is to write the comma-separated table to the console.
...	Arguments to write.csv. Used when *csv.out* is TRUE.

Value

Invisibly returns *x*.

Author(s)

Mikko Korpela

References

This function is based on the Fortran program REDFIT, which is in the public domain.

Schulz, M. and Mudelsee, M. (2002) REDFIT: estimating red-noise spectra directly from unevenly spaced paleoclimatic time series. *Computers & Geosciences*, **28**(3), 421–426.

See Also

redfit

Examples

```
library(utils)
data(ca533)
tm <- time(ca533)
x <- ca533[[1]]
idx <- which(!is.na(x))
redf <- redfit(x[idx], tm[idx], "time",
               nsim = 100, iwin = 0, ofac = 1, n50 = 1)
print(redf)
fname <- tempfile(fileext=".csv")
print(fname) # tempfile used for output
print(redf, csv.out = TRUE, file = fname)
redftable <- read.csv(fname)
unlink(fname) # remove the file
```

`print.rwl.report` *Do some reporting on a RWL object*

Description

This function prints the results of `rwl.report`

Usage

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

Arguments

`x`	a `list` from `rwl.report`
`...`	not implemented

Details

This function formats the `list` from `rwl.report` for the user to have a summary report of the number of series, the mean length of all the series, the first year, last year, the mean first-order autocorrelation (via `summary.rwl`), the mean interseries correlation (via `interseries.cor`), the years where a series has a missing ring (zero), internal NA, or a very small ring (<0.005).

Value

Invisible

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

rwl.report, summary.rwl, interseries.cor

Examples

```
data("gp.rwl")
rwl.report(gp.rwl)
foo <- gp.rwl
foo[177,1] <- NA
foo[177:180,3] <- NA
foo[185,4] <- 0.001
rwl.report(foo)
```

---

rasterPlot *Add Raster Elements to Plot*

Description

This function takes plotting commands and uses a temporary bitmap graphics device to capture their output. The resulting raster image is drawn in the plot or figure region of the active high-level plot. A new plot is started if one does not exist.

Usage

```
rasterPlot(expr, res = 150, region = c("plot", "figure"), antialias,
           bg = "transparent", interpolate = TRUE, draw = TRUE,
           Cairo = FALSE, ...)
```

Arguments

expr	Low-level plotting commands (lines, points, text, ...) representing elements to be added to the current plot. A call or an expression.
res	Resolution in points per inch (ppi). A numeric value. Suggested values for different types of display media are given in compactPDF. For example, the default 150 ppi corresponds to "ebook" quality.
region	The function can draw in the "plot" region or the "figure" region which also includes "mar" margins (see par). If the drawing commands in *expr* contain no material for the margins, the default "plot" is optimal. Plotting in the outer margins ("oma" in par) is not supported.
antialias	Antialiasing argument passed to png. The default (missing argument) is probably good for line plots but "none" is preferred for images in which color signifies value of data. Unused if a Cairo device is used instead of png.
bg	Background color of the raster plot, an argument passed to the bitmap device. If the default "transparent" does not work, try "white" or another color. Note that a non-transparent background will mask any previous content in the figure or plot region, depending on the value of *region*.

interpolate	Argument passed to rasterImage. A logical flag. The default is TRUE: use linear interpolation. Analogously to *antialias*, FALSE is preferred when color maps to value.
draw	A logical flag. Draw the results (TRUE, the default) or return an image object (FALSE)?
Cairo	A logical flag. TRUE for preferring a Cairo to png as the bitmap device, FALSE (the default) for the opposite. If the preferred device cannot be used, the other one will be tried.
...	Other arguments to png or Cairo.

Details

The appropriate graphical parameters of the current graphics device are copied to the temporary bitmap device. Therefore the appearance of the raster contents should be almost the same as when directly drawn.

The call or expression *expr* is evaluated in the environment of the caller.

It is possible that the raster contents will maintain a constant size when the graphics device is resized. If resizing works, however, the image may become distorted. For example, circle symbols will turn into ellipses if the width to height ratio is not maintained (see 'Examples'). This is in contrast to a standard plot in a display graphics device, e.g. x11, where text and symbols maintain their size when the device is resized.

Value

If *draw* is TRUE, there is no return value. The function is used for the side effects.

If *draw* is FALSE, an object of class "nativeRaster" is returned. The object can be used as input for rasterImage or grid.raster. See readPNG. If no bitmap device is available (see 'Note'), NULL is returned.

Note

- The graphics device used for the output must have support for including raster images. See "rasterImage" in dev.capabilities.
- The R build must have a functional png device, which requires one of the following capabilities: "png", "aqua" or "cairo". Alternatively, a Cairo device from package Cairo must be available with Cairo.capabilities "raster" or "png".

If either of these requirements is not met, at least one message is generated and the function reverts to regular plotting. The *bg* argument is then handled by drawing a filled rectangle. Also *region* is honored, but the other settings do not apply.

Author(s)

Mikko Korpela

Examples

```
library(graphics)
library(stats)

## Picture with various graphical elements
x <- 1:100
y0 <- quote(sin(pi * x / 20) + x / 100 + rnorm(100, 0, 0.2))
y <- eval(y0)
ylab <- deparse(y0)
spl <- smooth.spline(y)
plot(x, y, type = "n", axes = FALSE, ylab = ylab)
usr <- par("usr")
xrange <- usr[2] - usr[1]
xsize <- xrange * 0.4
nsteps <- 8
xmar <- xsize / 20
yrange <- usr[4] - usr[3]
ysize <- yrange / 20
ymar <- 0.5 * ysize
X <- seq(usr[1] + xmar, by = xsize / nsteps, length.out = nsteps + 1)
xleft <- X[-(nsteps + 1)]
xright <- X[-1]
pin <- par("pin")
maxrad <- xsize / 3 * min(1, pin[2] / pin[1])
nrad <- 16
minrad <- maxrad / nrad
Rad <- seq(maxrad, by = (minrad - maxrad) / (nrad - 1), length.out=nrad)
xmar2 <- xmar + maxrad
ymar2 <- (xmar2 / xrange) * pin[1] / pin[2] * yrange
expr <- quote({
    rect(xleft, usr[4] - 1.5 * ysize, xright, usr[4] - ymar,
         col = rainbow(8), border = NA)
    symbols(rep(usr[2] - xmar2, nrad), rep(usr[3] + ymar2, nrad),
            circles = Rad, inches = FALSE, add = TRUE, fg = NA,
            bg = gray.colors(nrad + 1, 1, 0)[-1])
    points(y)
    lines(spl)
})
rasterPlot(expr, res = 50)
box()
axis(1)
axis(2)

## The same picture with higher resolution but no antialiasing
plot(y, type = "n", axes = FALSE, ann = FALSE)
## No content in margin, but region = "figure" and bg = "white"
## paints margin white
rasterPlot(expr, antialias = "none", interpolate = FALSE,
           region = "figure", bg = "white")
## Draw box, axes, labels
parnew <- par(new = TRUE)
plot(x, y, type = "n", ylab = ylab)
```

```
par(parnew)

## Draw plot(1:5) with adjusted margins and additional axes.  Some parts
## are drawn with rasterPlot, others normally.  Resize to see stretching.
op <- par(no.readonly = TRUE)
par(mar = c(5.1, 4.1, 2.1, 2.1))
plot(1:5, type = "n", axes = FALSE, ann = FALSE)
expr2 <- quote({
    points(c(2, 4), c(2, 4))
    axis(2)
    axis(3)
})
rasterPlot(expr2, region = "figure", bg = "white")
points(c(1, 3, 5), c(1, 3, 5))
box()
axis(1)
axis(4)
title(xlab = "Index", ylab = "1:5")
par(op)
```

`rcs` *Regional Curve Standardization*

Description

Detrend multiple ring-width series simultaneously using a regional curve.

Usage

```
rcs(rwl, po, nyrs = NULL, f = 0.5, biweight = TRUE, ratios = TRUE,
    rc.out = FALSE, make.plot = TRUE, ..., rc.in = NULL, check = TRUE)
```

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`
`po`	a `data.frame` containing two variables. Variable one (*series* in the example below) gives the series ID as either `characters` or `factors`. These must exactly match `colnames(`*rwl*`)`. Variable two (*pith.offset* in the example below) must be integral values and give the years from the beginning of the core to the pith (or center) of the tree. The minimum value is 1.
`nyrs`	a number giving the rigidity of the smoothing spline, defaults to 0.1 of length of the maximum cambial age (i.e., the length of the regional curve) if *nyrs* is `NULL`.
`f`	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
`biweight`	`logical` flag. If `TRUE` then a robust mean is calculated using `tbrm`.
`ratios`	`logical` flag. If `TRUE` (the default) then indices are calculated by division, if `FALSE` indices are calculated by subtraction.

rc.out	logical flag. Returns the regional curve along with the ring-width indices if TRUE.
make.plot	logical flag. Makes plots of the raw data and regional curve if TRUE.
...	other arguments passed to plot.
rc.in	for internal use.
check	a logical flag. Bypass input checks by setting this to FALSE.

Details

This method detrends and standardizes tree-ring series by calculating an age-related growth curve specific to the *rwl*. The detrending is the estimation and removal of the tree's natural biological growth trend. The standardization is done by either dividing each series by the growth trend or subtracting the growth trend from each series to produce units in the dimensionless ring-width index (RWI). The option to produce indices by subtraction is intended to be used on series that have been subject to variance stabilization (e.g., using powt).

The spline approach uses an n-year spline where the frequency response is 0.50 at a wavelength of 10 percent of the maximum cambial age unless specified differently using *nyrs* and *f* in the function caps.

This attempts to remove the low frequency variability that is due to biological or stand effects. See the references below for further details on detrending in general, and Biondi and Qeadan (2008) for an explanation of RCS.

Value

A data.frame containing the dimensionless and detrended ring-width indices with column names, row names and dimensions of *rwl*. If *rc.out* is TRUE then a list will be returned with a data.frame containing the detrended ring widths as above and a vector containing the regional curve.

Note

DendroLab website: https://dendrolaborg.wordpress.com/

Author(s)

Code provided by DendroLab based on programming by F. Qeadan and F. Biondi, University of Nevada Reno, USA and adapted for dplR by Andy Bunn. Patched and improved by Mikko Korpela.

References

Biondi, F. and Qeadan, F. (2008) A theory-driven approach to tree-ring standardization: Defining the biological trend from expected basal area increment. *Tree-Ring Research*, **64**(2), 81–96.

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Fritts, H. C. (2001) *Tree Rings and Climate*. Blackburn. ISBN-13: 978-1-930665-39-2.

See Also

detrend, chron, cms, caps

Examples

```
library(utils)
data(gp.rwl)
data(gp.po)
gp.rwi <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
              rc.out = TRUE, make.plot = FALSE)
str(gp.rwi)
gp.rwi <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
              make.plot = TRUE, main = "Regional Curve")
```

`read.compact` *Read* DPL *Compact Format Ring Width File*

Description

This function reads in a DPL compact format file of ring widths.

Usage

```
read.compact(fname)
```

Arguments

`fname`	a `character` vector giving the file name of the rwl file.

Details

This function should be able to read files written by the Dendrochronology Program Library (DPL) in its compact format.

Value

An object of class `c("rwl", "data.frame")` with the series in columns and the years as rows. The series IDs are the column names and the years are the row names.

Author(s)

Mikko Korpela

See Also

`read.rwl`, `read.tucson`, `read.tridas`, `read.fh`, `write.compact`

Examples

```
## Not run:
data(co021)
fname <- write.compact(rwl.df = co021,
                       fname = tempfile(fileext=".rwl"),
                       append = FALSE, prec = 0.001)
foo <- read.compact(fname)
str(foo)
str(co021)
all.equal(foo,co021)
unlink(fname)

## End(Not run)
```

`read.crn` *Read Tucson Format Chronology File*

Description

This function reads in a Tucson (decadal) format file of tree-ring chronologies (.crn).

Usage

```
read.crn(fname, header = NULL, encoding = getOption("encoding"),
         long = TRUE)
```

Arguments

`fname`	a `character` vector giving the file name of the crn file.
`header`	`logical` flag indicating whether the file has a header. If `NULL` then the function will attempt to determine if a header exists
`encoding`	the name of the encoding to be used when reading the crn file. Usually the default value will work, but a crn file written in a non-default encoding may crash the function. In that case, identifying the encoding and specifying it here should fix the problem. Examples of popular encodings available on many systems are `"ASCII"`, `"UTF-8"`, and `"latin1"` alias `"ISO-8859-1"`. See the help of `file`.
`long`	`logical` flag indicating whether to automatically detect when an input file uses more than 4 characters for the decade. If `FALSE`, the function assumes 6 characters are used for the site ID and 4 characters for the decade, which is the standard. If `TRUE` (the default), long records may work.

Details

This reads in a standard crn file as defined according to the standards of the ITRDB at https://www1.ncdc.noaa.gov/pub/data/paleo/treering/treeinfo.txt. Despite the standards at the ITRDB, this occasionally fails due to formatting problems.

Value

A data.frame with each chronology in columns and the years as rows. The chronology IDs are the column names and the years are the row names. If the file includes sample depth that is included as the last column (*samp.depth*). The output class is class "crn" and "data.frame"

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

---

read.fh *Read Heidelberg Format Ring Width File*

Description

This function reads in a Heidelberg (block or column) format file of ring widths (.fh).

Usage

```
read.fh(fname, BC_correction = FALSE)
```

Arguments

fname	a character vector giving the file name of the fh file.
BC_correction	a logical. If TRUE the correction moves BC-dates one year forward.

Details

This reads in a fh-file with ring widths in blocks (decadal format) or in columns (e.g., as with comment flags) as used by TSAP program. Chronologies or half-chronos in fh-format are not supported.

Value

An object of class c("rwl", "data.frame") with the series in columns and the years as rows. The keycodes are the column names and the years are the row names. Depending on metadata available in the input file, the following attributes may be present in the data.frame:

ids	A data.frame identifying the series. Always contains columns "tree" and "core", may contain columns "site", "radius", and "stemDisk". All columns are numeric.
po	A data.frame containing pith offsets, formatted for use in rcs.

Author(s)

Christian Zang. New features and patches by Mikko Korpela and Ronald Visser.

References

Rinn, F. (2003) *TSAP-Win User Reference Manual.* Rinntech, Heidelberg. https://rinntech.info/products/tsap-win/.

See Also

read.rwl

---

read.ids *Read Site-Tree-Core* IDs

Description

These functions try to read site, tree, and core IDs from a rwl data.frame.

Usage

```
read.ids(rwl, stc = c(3, 2, 3), ignore.site.case = FALSE,
         ignore.case = FALSE, fix.typos = FALSE, typo.ratio = 5,
         use.cor = TRUE)

autoread.ids(rwl, ignore.site.case = TRUE, ignore.case = "auto",
             fix.typos = TRUE, typo.ratio = 5, use.cor = TRUE)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl or ca533
stc	a vector of three integral values or character string "auto". The numbers indicate the number of characters to split the site code (*stc*[1]), the tree IDs (*stc*[2]), and the core IDs (*stc*[3]). Defaults to c(3, 2, 3). If "auto", tries to automatically determine the split locations. See Details for further information.
use.cor	a logical flag. If TRUE and *stc* is "auto", correlation clustering may be used for determining the length of the tree and core parts. See Details.

The following parameters affect the handling of suspected typing errors. Some have different default values in read.ids and autoread.ids.

ignore.site.case	a logical flag. If TRUE, the function does not distinguish between upper case and lower case letters in the site part of the series names.
ignore.case	a logical flag or "auto". If TRUE, the function does not distinguish between upper case and lower case letters in the tree / core part of the series names. The default in read.ids is FALSE, i.e. the difference matters. The default in read.ids is "auto", which means that the function tries to be smart with respect to case sensitivity. In "auto" mode, the function generally ignores case differences, unless doing so would result in additional duplicate combinations

of tree and core IDs. Also, when in "auto" mode and *stc* is "auto", case sensitivity is used in highly heuristic ways when deciding the boundary between the site part and the tree part in uncertain cases.

`fix.typos` a logical flag. If TRUE, the function will try to detect and fix typing errors.

`typo.ratio` a numeric value larger than 1, affecting the eagerness of the function to fix typing errors. The default is 5. See Details.

Details

Because dendrochronologists often take more than one core per tree, it is occasionally useful to calculate within vs. between tree variance. The International Tree Ring Data Bank (ITRDB) allows the first eight characters in an rwl file for series IDs but these are often shorter. Typically the creators of rwl files use a logical labeling method that can allow the user to determine the tree and core ID from the label.

Argument *stc* tells how each series separate into site, tree, and core IDs. For instance a series code might be "ABC011" indicating site "ABC", tree 1, core 1. If this format is consistent then the *stc* mask would be c(3, 2, 3) allowing up to three characters for the core ID (i.e., pad to the right). If it is not possible to define the scheme (and often it is not possible to machine read IDs), then the output data.frame can be built manually. See Value for format.

The function autoread.ids is a wrapper to read.ids with stc="auto", i.e. automatic detection of the site / tree / core scheme, and different default values of some parameters. In automatic mode, the names in the same *rwl* can even follow different site / tree / core schemes. As there are numerous possible encoding schemes for naming measurement series, the function cannot always produce the correct result.

With stc="auto", the site part can be one of the following.

- In names mostly consisting of numbers, the longest common prefix is the site part
- Alphanumeric site part ending with alphabet, when followed by numbers and alphabets
- Alphabetic site part (quite complicated actual definition). Setting *ignore.case* to "auto" allows the function to try to guess when a case change in the middle of a sequence of alphabets signifies a boundary between the site part and the tree part.
- The characters before the first sequence of space / punctuation characters in a name that contains at least two such sequences

These descriptions are somewhat general, and the details can be found in regular expressions inside the function. If a name does not match any of the descriptions, it is matched against a previously found site part, starting from the longest.

The following ID schemes are detected and supported in the tree / core part. The detection is done per site.

- Numbers in tree part, core part starts with something else
- Alphabets in tree part, core part starts with something else
- Alphabets, either tree part all lower case and core part all upper case or vice versa. For this to work, *ignore.case* must be set to "auto" or FALSE.
- All digits. In this case, the number of characters belonging to the tree and core parts is detected with one of the following methods.

  - If numeric tree parts were found before, it is assumed that the core part is missing (one core per tree).
  - It the series are numbered continuously, one core per tree is assumed.
  - Otherwise, try to find a core part as the suffix so that the cores are numbered continuously.

  If none of the above fits, the tree / core split of the all-digit names will be decided with the methods described further down the list, or finally with the fallback mechanism.

- The combined tree / core part is empty or one character. In this case, the core part is assumed to be missing.
- Tree and core parts separated by a punctuation or white space character

If the split of a tree / core part cannot be found with any of the methods described above, the prefix of the string is matched against a previously found tree part, starting from the longest. The fallback mechanism for the still undecided tree / core parts is one of the following. The first one is used if *use.cor* is `TRUE`, number two if it is `FALSE`.

1. Pairwise correlation coefficients are computed between all remaining series. Pairs of series with above median correlation are flagged as similar, and the other pairs are flagged as dissimilar. Each possible number of characters (minimum 1) is considered for the share of the tree ID. The corresponding unique would-be tree IDs determine a set of clusterings where one cluster is formed by all the measurement series of a single tree. For each clustering (allocation of characters), an agreement score is computed. The agreement score is defined as the sum of the number of similar pairs with matching cluster number and the number of dissimilar pairs with non-matching cluster number. The number of characters with the maximum agreement is chosen.
2. If the majority of the names in the site use k characters for the tree part, that number is chosen. Otherwise, one core per tree is assumed. Parameter *typo.ratio* has a double meaning as it also defines what is meant by majority here: at least *typo.ratio* / (*typo.ratio* + 1) $*$ *n.tot*, where *n.tot* is the number of names in the site.

In both fallback mechanisms, the number of characters allocated for the tree part will be increased until all trees have a non-zero ID or there are no more characters.

Suspected typing errors will be fixed by the function if *fix.typos* is `TRUE`. The parameter *typo.ratio* affects the eagerness to fix typos, i.e. the number of counterexamples required to declare a typo. The following main typo fixing mechanisms are implemented:

Site IDs. If a rare site string resembles an at least *typo.ratio* times more frequent alternative, and if fixing it would not create any name collisions, make the fix. The alternative string must be unique, or if there is more than one alternative, it is enough if only one of them is a look-alike string. Any kind of substitution in one character place is allowed if the alternative string has the same length as the original string. The alternative string can be one character longer or one character shorter than the original string, but only if it involves interpreting one digit as the look-alike alphabet or vice versa. There are requirements to how long a site string must be in order to be eligible for replacement / typo fixing, i.e. cannot be shortened to zero length, cannot change the only character of a site string. The parameters *ignore.case* and *ignore.site.case* have some effect on this typo fixing mechanism.

Tree and core IDs. If all tree / core parts of a site have the same length, each character position is inspected individually. If the characters in the i:th position are predominantly digits (alphabets), any alphabets (digits) are changed to the corresponding look-alike digit (alphabet) if

there is one. The look-alike groups are {0, O, o}, {1, I, i}, {5, S, s} and {6, G}. The parameter *typo.ratio* determines the decision threshold of interpreting the type of each character position as alphabet (digit): the ratio of alphabets (digits) to the total number of characters must be at least *typo.ratio* / (*typo.ratio* + 1). If a name differs from the majority type in more than one character position, it is not fixed. Also, no fixes are performed if any of them would cause a possible monotonic order of numeric prefixes to break.

The function attempts to convert the tree and core substrings to integral values. When this succeeds, the converted values are copied to the output without modification. When non-integral substrings are observed, each unique tree is assigned a unique integral value. The same applies to cores within a tree, but there are some subtleties with respect to the handling of duplicates. Substrings are sorted before assigning the numeric IDs.

The order of columns in *rwl*, in most cases, does not affect the tree and core IDs assigned to each series.

Value

A data.frame with column one named "tree" giving an ID for each tree and column two named "core" giving an ID for each core. The original series IDs are copied from rwl as rownames. The order of the rows in the output matches the order of the series in *rwl*. If more than one site is detected, an additional third column named "site" will contain a site ID. All columns have integral valued numeric values.

Author(s)

Andy Bunn (original version) and Mikko Korpela (patches, *stc*="auto", *fix.typos*, etc.).

See Also

rwi.stats, read.rwl

Examples

```
library(utils)
data(ca533)
read.ids(ca533, stc = c(3, 2, 3))
autoread.ids(ca533)
```

---

read.rwl *Read Ring Width File*

Description

This function reads in a file of ring widths (.rwl) in one of the available formats.

Usage

```
read.rwl(fname,
        format = c("auto", "tucson", "compact", "tridas", "heidelberg", "csv"),
         ...)
```

Arguments

fname	a character vector giving the file name of the rwl file.
format	a character vector giving the format. This must be "auto" (rough automatic detection), "tucson", "compact", "tridas", "heidelberg" or "csv". Automatic format detection is the default but failable.
...	arguments specific to the function implementing the operation for the chosen format.

Details

This is a simple wrapper to the functions actually implementing the read operation.

Value

If a "tucson", "compact", "heidelberg", "csv" file is read (even through "auto"), returns an object of class c("rwl", "data.frame") with the series in columns and the years as rows. The series IDs are the column names and the years are the row names.

If a "tridas" file is read (even through "auto"), returns a list of results. See read.tridas for more information.

Author(s)

Mikko Korpela

See Also

read.tucson, read.compact, read.tridas, read.fh, csv2rwl, write.rwl

---

read.tridas *Read Tree Ring Data Standard (TRiDaS) File*

Description

This function reads in a TRiDaS format XML file. Measurements, derived series and various kinds of metadata are supported.

Usage

```
read.tridas(fname, ids.from.titles = FALSE,
            ids.from.identifiers = TRUE, combine.series = TRUE,
            trim.whitespace = TRUE, warn.units = TRUE)
```

Arguments

fname	character vector giving the file name of the TRiDaS file.
ids.from.titles	logical flag indicating whether to override the (tree, core, radius, measurement) structure imposed by the element hierarchy (element, sample, radius, measurementSeries) of the file. If TRUE, measurement series will be rearranged by matching titles in the file at the aforementioned four levels of the hierarchy. Defaults to FALSE, i.e. the element hierarchy of the file will be used.
ids.from.identifiers	logical flag indicating whether to (partially) override the element hierarchy of the file. If TRUE, measurement series will be grouped according to matching identifiers at the measurementSeries level, where identifiers are available. The changes caused by this option are applied on top of the structure imposed by the file or computed from matching titles if *ids.from.titles* == TRUE. Defaults to TRUE.
combine.series	logical flag indicating whether to combine two or more measurement series with the same set of (tree, core, radius, measurement) ID numbers. Each set of combined measurement series will be represented by one column of a resulting data.frame. Overlapping years of combined series do not produce a warning. If several data points are available for a given year, the function chooses one in a rather arbitrary manner. This option can only have effect when *ids.from.titles* || *ids.from.identifiers*.
trim.whitespace	logical flag indicating whether to replace repeated white spaces in the text content of the file with only one space. Defaults to TRUE, i.e. excess white space will be trimmed from the text.
warn.units	logical flag indicating whether to warn about unitless measurements and "strange" units. The function expects measurements in units that can be converted to millimetres. Defaults to TRUE: warnings will be given. For example, density measurements will trigger warnings, which can be disabled by setting this option to FALSE.

Details

The Tree Ring Data Standard (TRiDaS) is described in Jansma et. al (2010).

The parameters used for rearranging (*ids.from.titles*, *ids.from.identifiers*) and combining (*combine.series*) measurement series only affect the four lowest levels of document structure: element, sample, radius, measurementSeries. Series are not reorganized or combined at the upper structural levels (project, object).

Value

A list with a variable number of components according to the contents of the input file. The possible list components are:

measurements	A data.frame or a list of data.frames with the series in columns and the years as rows. Contains measurements ('<measurementSeries>') with known years.

The series IDs are the column names and the years are the row names. The series IDs are derived from '<title>' elements in the input file. Each unique combination of '<project>', '<object>', '<unit>', '<taxon>', and '<variable>' gets a separate data.frame.

ids — A data.frame or a list of data.frames with columns named "tree", "core", "radius", and "measurement", together giving a unique numeric ID for each column of the data.frame(s) in *measurements*.

If !*combine.series* && (*ids.from.titles* || *ids.from.identifiers*), some rows may be non-unique.

titles — A data.frame or a list of data.frames with columns named "tree", "core", "radius", and "measurement", containing the '<title>' hierarchy of each column of the data.frame(s) in *measurements*.

wood.completeness — A data.frame or a list of data.frames containing wood completeness information. Column names are a subset of the following, almost self-explanatory set: "pith.presence", "heartwood.presence", "sapwood.presence", "last.ring.presence", "last.ring.details", "bark.presence", "n.sapwood", "n.missing.heartwood", "n.missing.sapwood", "missing.heartwood.foundation", "missing.sapwood.foundation", "n.unmeasured.inner", "n.unmeasured.outer".

unit — A character vector giving the unit of the measurements. Length equals the number of data.frames in *measurements*.

project.id — A numeric vector giving the project ID, i.e. the position of the corresponding '<project>' element), of the measurements in each data.frame in *measurements*. Length equals the number of data.frames.

project.title — A character vector giving the title of the project of each data.frame in *measurements*. Length equals the number of data.frames.

site.id — A data.frame giving the site ID (position of '<object>' element(s) within a '<project>') of each data.frame in *measurements*. May have several columns to reflect the possibly nested '<object>' elements.

site.title — A data.frame giving the site ('<object>') title of each data.frame in *measurements*. May have several columns to reflect the possibly nested '<object>' elements.

taxon — A data.frame showing the taxonomic name for each data.frame in *measurements*. Contains some of the following columns: "text", "lang", "normal", "normalId", "normalStd". The first two are a free-form name and its language, and the rest are related to a normalized name.

variable — A data.frame showing the measured variable of each data.frame in *measurements*. Contains some of the following columns: "text", "lang", "normal", "normalId", "normalStd", "normalTridas". The first two are a free-form name and its language, and the rest are related to a normalized name.

undated — A list of measurements with unknown years, together with metadata. Elements are a subset of the following:

data A numeric vector or a list of such vectors containing measurement series

unit A character vector giving the unit of the measurements. Length equals the number of measurement series in *undated*$*data*

ids A `data.frame` with columns named `"tree"`, `"core"`, `"radius"`, and `"measurement"`, together giving a `numeric` ID for each measurement series in *undated*$*data*. The rows are guaranteed to be unique only when comparing measurement series with the same *project.id* and *site.id*, but not if *ids.from.titles* || *ids.from.identifiers*.

titles A `data.frame` with columns named `"tree"`, `"core"`, `"radius"`, and `"measurement"`, containing the '`<title>`' hierarchy of each measurement series in *undated*$*data*

project.id A `numeric` vector giving the project ID of each measurement series in *undated*$*data*

project.title A `character` vector giving the project title of each measurement series in *undated*$*data*

site.id A `data.frame` giving the site ID of each measurement series in *undated*$*data*

site.title A `data.frame` giving the site title of each measurement series in *undated*$*data*

variable A `data.frame` containing the variable of each measurement series in *undated*$*data*

taxon A `data.frame` containing taxonomic names of each measurement series in *undated*$*data*

wood.completeness A `data.frame` containing wood completeness information of each measurement series in *undated*$*data*

`derived` A `list` of calculated series of values, together with metadata. Elements are a subset of the following:

data A `numeric` vector or a list of such vectors containing calculated series of values.

link A `list` of `data.frames`, one for each series in *derived*$*data*, giving links to the measurements used to form the corresponding derived series. Each `data.frame` has a subset of the following columns: `"idRef"` (reference to a series in the same file), `"xLink"` (URI), `"identifier"`, and `"domain"` (identifier and its domain, not necessarily in the same file).

project.id A `numeric` vector giving the project ID of each derived series in *derived*$*data*

id A `numeric` vector giving the ID (order of appearance in the project) of each derived series in *derived*$*data*

title A `character` vector giving the title of each derived series in *derived*$*data*

project.title A `character` vector giving the project title of each derived series in *derived*$*data*

unit A `character` vector giving the unit of the derived series. Length equals the number of series in *derived*$*data*.

standardizing.method A `character` vector giving the standardizing method of the derived series. Length equals the number of series in *derived*$*data*.

variable A `data.frame` containing the variable of each series in *derived*$*data*

`type` A `data.frame` containing the type of various entities, and metadata related to each '`type`' element. Contents are `NA` where the metadata is not applicable (e.g., no *tree.id* when the '`type`' element refers to a project). Columns are a subset of the following:

text The text of the 'type' element

lang The language of the text

normal The normalized name of the type

normalId The ID value of the type in the standard dictionary

normalStd The name of the standard

project.id The ID of the project

site.id One or more columns with this prefix, depending on the maximum depth of the '<object>' hierarchy. Gives the ID of the site where the '<type>' element appeared.

tree.id The ID of the tree

core.id The ID of the core

derived.id The ID of the derived series

project.title The title of the project

site.title One or more columns with this prefix, depending on the maximum depth of the '<object>' hierarchy. Gives the title of the site where the '<type>' element appeared.

tree.title The title of the tree

core.title The title of the core

derived.title The title of the derived series

`comments` A `data.frame` containing comments to various entities, and metadata related to each 'comments' element. Contents are NA where the metadata is not applicable. Columns are a subset of the following:

text The text of the 'comments' element

project.id The ID of the project

site.id One or more columns with this prefix, depending on the maximum depth of the '<object>' hierarchy. Gives the ID of the site.

tree.id The ID of the tree

core.id The ID of the core

radius.id The ID of the radius

measurement.id The ID of the measurement series

derived.id The ID of the derived series

project.title The title of the project

site.title One or more columns with this prefix, depending on the maximum depth of the '<object>' hierarchy. Gives the title of the site.

tree.title The title of the tree

core.title The title of the core

radius.title The title of the radius

measurement.title The title of the measurement series

derived.title The title of the derived series

`identifier` A `data.frame` containing identifiers of various entities, and metadata related to each 'identifier' element. Contents are NA where the metadata is not applicable. Columns are a subset of the following:

text The text of the 'identifier' element

domain The domain which the identifier is applicable to

project.id The ID of the project

site.id One or more columns with this prefix, depending on the maximum depth of the '<object>' hierarchy. Gives the ID of the site.

tree.id The ID of the tree

core.id The ID of the core

radius.id The ID of the radius

measurement.id The ID of the measurement series

derived.id The ID of the derived series

project.title The title of the project

site.title One or more columns with this prefix, depending on the maximum depth of the '<object>' hierarchy. Gives the title of the site.

tree.title The title of the tree

core.title The title of the core

radius.title The title of the radius

measurement.title The title of the measurement series

derived.title The title of the derived series

`remark` A `list` of remarks concerning individual measured or derived values, with some of the following items:

measurements Remarks related to measurements with a known year. A `data.frame` with the following columns:

text The remark

frame Index to a `data.frame` in *measurements*

row Index to a row of the `data.frame`

col Index to a column of the `data.frame`

undated Remarks related to measurements without a known year. A `data.frame` with the following columns:

text The remark

series Index to a series in *undated*$*data*

idx Index to a value in the series

derived Remarks related to derived values. A `data.frame` with the following columns:

text The remark

series Index to a series in *derived*$*data*

idx Index to a value in the series

`laboratory` A `data.frame` or a `list` of `data.frames` with one item per project. Each `data.frame` contains information about the research laboratories involved in the project. Columns are a subset of the following:

name Name of the laboratory

acronym Acronym of the name

identifier Identifier

domain Domain which the identifier is applicable to

addressLine1 Address

addressLine2 Another address line

cityOrTown City or town

stateProvinceRegion State, province or region

postalCode Postal code

country Country

`research` A `data.frame` or a `list` of `data.frames` with one item per project. Each `data.frame` contains information about the systems in which the research project is registered. Columns are the following:

identifier Identifier

domain Domain which the identifier is applicable to

description General description

`altitude` A `data.frame` containing the altitude of trees. Columns are the following:

metres The altitude in metres

project.id The ID of the project

site.id One or more columns with this prefix, depending on the maximum depth of the '`<object>`' hierarchy. Gives the ID of the site.

tree.id The ID of the tree

project.title The title of the project

site.title One or more columns with this prefix, depending on the maximum depth of the '`<object>`' hierarchy. Gives the title of the site.

tree.title The title of the tree

`preferred` A `data.frame` containing links to preferred measurement series. Columns are a subset of the following:

idRef Reference to a series in the same file

xLink URI

identifier Identifier of a series not necessarily in the same file

domain Domain which the identifier is applicable to

project.id The ID of the project

site.id One or more columns with this prefix, depending on the maximum depth of the '`<object>`' hierarchy. Gives the ID of the site.

tree.id The ID of the tree

project.title The title of the project

site.title One or more columns with this prefix, depending on the maximum depth of the '`<object>`' hierarchy. Gives the title of the site.

tree.title The title of the tree

Note

This is an early version of the function. Bugs are likely to exist, and parameters and return values are subject to change. Not all metadata defined in the TRiDaS specification is supported – unsupported elements are quietly ignored.

Author(s)

Mikko Korpela

References

Jansma, E., Brewer, P. W., and Zandhuis, I. (2010) TRiDaS 1.1: The tree-ring data standard. *Dendrochronologia*, **28**(2), 99–130.

See Also

`read.rwl`, `read.tucson`, `read.compact`, `read.fh`, `write.tridas`

`read.tucson` *Read Tucson Format Ring Width File*

Description

This function reads in a Tucson (decadal) format file of ring widths (.rwl).

Usage

```
read.tucson(fname, header = NULL, long = FALSE,
            encoding = getOption("encoding"),
            edge.zeros = TRUE, verbose = TRUE)
```

Arguments

`fname`	a `character` vector giving the file name of the rwl file.
`header`	`logical` flag indicating whether the file has a header. If `NULL` then the function will attempt to determine if a header exists.
`long`	`logical` flag indicating whether dates in file span 0 CE and therefore use negative numbers. If `TRUE` only the first 7 characters can be used for series IDs. If `FALSE` then series IDs can be up to 8 characters.
`encoding`	the name of the encoding to be used when reading the rwl file. Usually the default value will work, but an rwl file written in a non-default encoding may crash the function. In that case, identifying the encoding and specifying it here should fix the problem. Examples of popular encodings available on many systems are `"ASCII"`, `"UTF-8"`, and `"latin1"` alias `"ISO-8859-1"`. See the help of `file`.
`edge.zeros`	`logical` flag indicating whether leading or trailing zeros in series will be preserved (when the flag is `TRUE`, the default) or discarded, i.e. marked as `NA` (when `FALSE`).
`verbose`	`logical` flag, print info on data.

Details

This reads in a standard rwl file as defined according to the standards of the ITRDB at https://www1.ncdc.noaa.gov/pub/data/paleo/treering/treeinfo.txt. Despite the standards at the ITRDB, this occasionally fails due to formatting problems.

Value

An object of class `c("rwl", "data.frame")` with the series in columns and the years as rows. The series IDs are the column names and the years are the row names.

Author(s)

Andy Bunn. Patched and greatly improved by Mikko Korpela.

See Also

`read.rwl`, `read.compact`, `read.tridas`, `read.fh`, `write.tucson`

---

redfit *Red-Noise Spectra of Time-Series*

Description

Estimate red-noise spectra from a possibly unevenly spaced time-series.

Usage

```
redfit(x, t, tType = c("time", "age"), nsim = 1000, mctest = TRUE,
       ofac = 4, hifac = 1, n50 = 3, rhopre = NULL,
       p = c(0.10, 0.05, 0.02), iwin = 2,
       txOrdered = FALSE, verbose = FALSE, seed = NULL,
       maxTime = 10, nLimit = 10000)

runcrit(n, p = c(0.10, 0.05, 0.02), maxTime = 10, nLimit = 10000)
```

Arguments

`x`	a `numeric` vector representing a possibly unevenly spaced time-series.
`t`	a `numeric` vector of times or ages corresponding to x. If missing, this is set to an evenly spaced increasing integer sequence $(1, 2, \ldots)$ along x. See *txOrdered*.
`tType`	a `character` string indicating the type of the t vector: either times or ages.
`nsim`	a `numeric` value giving the number of simulated AR1 spectra to compute.
`mctest`	a `logical` flag. If `TRUE`, performs a Monte Carlo test for computing red noise false-alarm levels. In that case, the result list will contain non-`NULL` elements `"ci80"`, `"ci90"`, `"ci95"` and `"ci99"`.
`ofac`	oversampling factor for Lomb-Scargle Fourier transform. A `numeric` value.

hifac	maximum frequency to analyze relative to the Nyquist frequency. A numeric value.
n50	number of segments. The segments overlap by about 50 percent.
rhopre	a numeric value giving the prescribed autocorrelation coefficient. If NULL or negative, the autocorrelation coefficient will be estimated from the data.
p	a numeric or bigq (if package "gmp" is installed) vector of significance levels for a statistical test considering the number of runs in a sequence. See 'Details'.
iwin	the type of window used for scaling the values of each segment of data. A numeric value or one of "rectangular", "welch i", "hanning", "triangular" and "blackman-harris". Integers 0:4 correspond to the character values, in this order. The default *iwin* = 2 corresponds to the "hanning" window.
txOrdered	a logical flag. If TRUE, it is assumed that t is in ascending order without duplicates. If FALSE (the default), t will be sorted and x reordered accordingly. Values of x at identical values of t are averaged. If duplicates are found, the non-duplicated t and averaged x will be included in the return value of the function.
verbose	a logical flag. If TRUE, the function prints some information while operating.
seed	a value to be given to set.seed at the start of the function. The value is also recorded in the list returned. If not NULL, this can be used for reproducible results.
maxTime	a numeric value giving the approximate maximum time in seconds to use for computing the exact acceptance regions of the number of runs test. See also *nLimit*.
nLimit	a numeric value giving the maximum n for which runcrit will try to compute an exact solution to the acceptance regions of the number of runs test. Precomputed solutions may exist for larger n). This limit is in place because a part of the exact solution always needs to be computed in order to roughly estimate the total time and whether it would exceed *maxTime*. If *nLimit* is very large, *maxTime* may be (greatly) exceeded while computing the aforementioned part of the exact solution.
n	an integral value giving the length of the sequence in the number of runs test.

Details

Function redfit computes the spectrum of a possibly unevenly sampled time-series by using the Lomb-Scargle Fourier transform. The spectrum is bias-corrected using spectra computed from simulated AR1 series and the theoretical AR1 spectrum.

The function duplicates the functionality of program REDFIT by Schulz and Mudelsee. See the manual of that program for more information. The results of this function should be very close to REDFIT. However, some changes have been made:

- More precision is used in some constants and computations.
- All the data are used: the last segment always contains the last pair of (t, x). There may be small differences between redfit and REDFIT with respect to the number of points per segment and the overlap of consecutive segments.

- The critical values of the runs test (see the description of `runcrit` below) differ between `redfit` and REDFIT. The approximate equations in REDFIT produce values quite far off from the exact values when the number of frequencies is large.
- The user can select the significance levels of the runs test.
- Most of the window functions have been adjusted.
- 6 dB bandwidths have been computed for discrete-time windows.

Function `runcrit` computes the limits of the acceptance region of a number of runs test: assuming a sequence of n i.i.d. discrete random variables with two possible values a and b of equal probability (0.5), we are examining the distribution of the number of runs. A run is an uninterrupted sequence of only a or only b. The minimum number of runs is 1 (a sequence with only a or only b) while the maximum number is n (alternating a and b). See Bradley, p. 253–254, 259–263. The function is also called from `redfit`; see *rcnt* in 'Value' for the interpretation. In this case the arguments p, *maxTime* and *nLimit* are passed from `redfit` to `runcrit`, and n is the number of output frequencies.

The results of `runcrit` have been essentially precomputed for some values of p and n. If a precomputed result is not found and n is not too large (*nLimit*, *maxTime*), the exact results are computed on-demand. Otherwise, or if package `"gmp"` is not installed, the normal distribution is used for approximation.

Value

Function `runcrit` returns a `list` containing *rcritlo*, *rcrithi* and *rcritexact* (see below). Function `redfit` returns a `list` with the following elements:

`varx`	variance of x estimated from its spectrum. A `numeric` value.
`rho`	average autocorrelation coefficient, either estimated from the data or prescribed (*rhopre*). A `numeric` value.
`tau`	the time scale of an AR1 process corresponding to *rho*. A `numeric` value.
`rcnt`	a `numeric` value giving the number of runs to be used for a statistical test studying the difference between a theoretical AR1 spectrum and the bias-corrected spectrum estimated from the data. Null hypothesis: the two spectra agree, i.e. the probability of either being larger than the other is 0.5 at every point. Requires that *iwin* `== 0` (`"rectangular"`), *ofac* `== 1` and *n50* `== 1`. Otherwise the test is not performed and *rcnt* is `NULL`.
`rcritlo`	a `numeric` vector of critical low values for the number of runs, i.e. the lowest value for accepting the null hypothesis at each level of significance p. When returned from `redfit`, `NULL` when *rcnt* is `NULL`.
`rcrithi`	a `numeric` vector of critical high values for the number of runs, i.e. the highest value for accepting the null hypothesis at each level of significance p. When returned from `redfit`, `NULL` when *rcnt* is `NULL`.
`rcritexact`	a `logical` vector specifying whether each pair of *rcritlo* and *rcrithi* are exact values (`TRUE`) or approximated from a normal distribution (`FALSE`). When returned from `redfit`, `NULL` when *rcnt* is `NULL`.
`freq`	the frequencies used. A `numeric` vector. The other numeric vectors have the same length, i.e. one value for each frequency studied.

`gxx` estimated spectrum of the data (t, x). A `numeric` vector.

`gxxc` red noise corrected spectrum of the data. A `numeric` vector.

`grravg` average AR1 spectrum over *nsim* simulations. A `numeric` vector.

`gredth` theoretical AR1 spectrum. A `numeric` vector.

`corr` a `numeric` vector containing the by-frequency correction: *gxxc* equals *gxx* divided by *corr* (or multiplied by the inverse correction). Also used for computing *ci80*, *ci90*, *ci95* and *ci99*.

`ci80` a `numeric` vector containing the bias-corrected 80th percentile (by frequency) red noise spectrum. Only if *mctest* is `TRUE`.

`ci90` a `numeric` vector containing the 90th percentile red noise spectrum.

`ci95` 95th percentile red noise spectrum.

`ci99` 99th percentile red noise spectrum.

`call` the call to the function. See `match.call`.

`params` A `list` with the following items:

np number of points in the data.

nseg number of points in each segment.

nfreq number of frequencies in the results.

avgdt average sampling interval.

df difference between consecutive frequencies.

fnyq average Nyquist frequency.

n50 value of the *n50* argument.

ofac value of the *ofac* argument.

hifac value of the *hifac* argument.

segskip the ideal, non-rounded difference between starting indices of consecutive segments.

iwin value of the *iwin* argument. If a `character` value was used, this contains the corresponding number used internally.

nsim value of the *nsim* argument.

mctest value of the *mctest* argument.

rhopre value of the *rhopre* argument.

`vers` version of dplR. See `packageVersion`.

`seed` value of the *seed* argument.

`t` if duplicated values of t are given, the non-duplicated `numeric` time or age vector (see *tType*) is returned here. Otherwise `NULL`. See *txOrdered*.

`x` if duplicated values of t are given, the averaged `numeric` data vector is returned here. Otherwise `NULL`.

Author(s)

Mikko Korpela. Examples by Andy Bunn.

References

Function `redfit` is based on the Fortran program REDFIT (version 3.8e), which is in the public domain.

Bradley, J. V. (1968) *Distribution-Free Statistical Tests*. Prentice-Hall.

Schulz, M. and Mudelsee, M. (2002) REDFIT: estimating red-noise spectra directly from unevenly spaced paleoclimatic time series. *Computers & Geosciences*, **28**(3), 421–426.

See Also

`print.redfit`

Examples

```
# Create a simulated tree-ring width series that has a red-noise
# background ar1=phi and sd=sigma and an embedded signal with
# a period of 10 and an amplitude of have the rednoise sd.
library(graphics)
library(stats)
runif(1)
rs <- .Random.seed
set.seed(123)
nyrs <- 500
yrs <- 1:nyrs

# Here is an ar1 time series with a mean of 2mm,
# an ar1 of phi, and sd of sigma
phi <- 0.7
sigma <- 0.3
sigma0 <- sqrt((1 - phi^2) * sigma^2)
x <- arima.sim(list(ar = phi), n = nyrs, sd = sigma0) + 2

# Here is a sine wave at f=0.1 to add in with an amplitude
# equal to half the sd of the red noise background
per <- 10
amp <- sigma0 / 2
wav <- amp * sin(2 * pi / per * yrs)

# Add them together so we have signal and noise
x <- x + wav

# Here is the redfit spec
redf.x <- redfit(x, nsim = 500)

# Acceptance region of number of runs test
# (not useful with default arguments of redfit())
runcrit(length(redf.x[["freq"]]))

op <- par(no.readonly = TRUE) # Save to reset on exit
par(tcl = 0.5, mar = rep(2.2, 4), mgp = c(1.1, 0.1, 0))

plot(redf.x[["freq"]], redf.x[["gxxc"]],
```

```
     ylim = range(redf.x[["ci99"]], redf.x[["gxxc"]]),
     type = "n", ylab = "Spectrum", xlab = "Frequency (1/yr)",
     axes = FALSE)
grid()
lines(redf.x[["freq"]], redf.x[["gxxc"]], col = "#1B9E77")
lines(redf.x[["freq"]], redf.x[["ci99"]], col = "#D95F02")
lines(redf.x[["freq"]], redf.x[["ci95"]], col = "#7570B3")
lines(redf.x[["freq"]], redf.x[["ci90"]], col = "#E7298A")
freqs <- pretty(redf.x[["freq"]])
pers <- round(1 / freqs, 2)
axis(1, at = freqs, labels = TRUE)
axis(3, at = freqs, labels = pers)
mtext(text = "Period (yr)", side = 3, line = 1.1)
axis(2); axis(4)
legend("topright", c("x", "CI99", "CI95", "CI90"), lwd = 2,
       col = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A"),
       bg = "white")
box()

## Not run:
# Second example with tree-ring data
# Note the long-term low-freq signal in the data. E.g.,
# crn.plot(cana157)

library(utils)
data(cana157)
yrs <- time(cana157)
x <- cana157[, 1]
redf.x <- redfit(x, nsim = 1000)

plot(redf.x[["freq"]], redf.x[["gxxc"]],
     ylim = range(redf.x[["ci99"]], redf.x[["gxxc"]]),
     type = "n", ylab = "Spectrum", xlab = "Frequency (1/yr)",
     axes = FALSE)
grid()
lines(redf.x[["freq"]], redf.x[["gxxc"]], col = "#1B9E77")
lines(redf.x[["freq"]], redf.x[["ci99"]], col = "#D95F02")
lines(redf.x[["freq"]], redf.x[["ci95"]], col = "#7570B3")
lines(redf.x[["freq"]], redf.x[["ci90"]], col = "#E7298A")
freqs <- pretty(redf.x[["freq"]])
pers <- round(1 / freqs, 2)
axis(1, at = freqs, labels = TRUE)
axis(3, at = freqs, labels = pers)
mtext(text = "Period (yr)", side = 3, line = 1.1)
axis(2); axis(4)
legend("topright", c("x", "CI99", "CI95", "CI90"), lwd = 2,
       col = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A"),
       bg = "white")
box()
par(op)

## End(Not run)
.Random.seed <- rs
```

`rwi.stats.running` *(Running Window) Statistics on Detrended Ring-Width Series*

Description

These functions calculate descriptive statistics on a `data.frame` of (usually) ring-width indices. The statistics are optionally computed in a running window with adjustable length and overlap. The data can be filtered so that the comparisons are made to on just high-frequency data.

Usage

```
rwi.stats.running(rwi, ids = NULL, period = c("max", "common"),
                  method = c("spearman", "pearson","kendall"),
                  prewhiten=FALSE,n=NULL,
                  running.window = TRUE,
                  window.length = min(50, nrow(rwi)),
                  window.overlap = floor(window.length / 2),
                  first.start = NULL,
                  min.corr.overlap = min(30, window.length),
                  round.decimals = 3,
                  zero.is.missing = TRUE)

rwi.stats(rwi, ids=NULL, period=c("max", "common"),
          method = c("spearman", "pearson","kendall"), ...)

rwi.stats.legacy(rwi, ids=NULL, period=c("max", "common"))
```

Arguments

`rwi`	a `data.frame` with detrended and standardized ring width indices as columns and years as rows such as that produced by `detrend`.
`ids`	an optional `data.frame` with column one named `"tree"` giving a `numeric` ID for each tree and column two named `"core"` giving a `numeric` ID for each core. Defaults to one core per tree as `data.frame(tree=1:ncol(`*rwi*`), core=rep(1, ncol(`*rwi*`)))`.
`period`	a `character` string, either `"common"` or `"max"` indicating whether correlations should be limited to complete observations over the period common to all cores (i.e. rows common to all samples) or the maximum pairwise overlap. Defaults to `"max"`.
`method`	Can be either `"pearson"`, `"kendall"`, or `"spearman"` which indicates the correlation coefficient to be used. Defaults to `"spearman"`. See `cor`.
`n`	`NULL` or an integral value giving the filter length for the `hanning` filter used for removal of low frequency variation.
`prewhiten`	`logical` flag. If `TRUE` each series is whitened using `ar`.

`running.window` `logical` flag indicating whether to use a running window (`TRUE`, the default) or to ignore the other window parameters and effectively use one window covering all years (`FALSE`).

`window.length` `numeric` value indicating the length of the running window in years. The default is 50 years or the number of years (rows) in *rwi*, whichever is smaller.

`window.overlap` `numeric` value indicating the overlap of consecutive window positions, i.e. the number of common years. The default is half of the window length, rounded down.

`first.start` an optional `numeric` value setting the position of the first window. Must be a value between 1 and *n.years*`-`*window.length*`+1`, where *n.years* is the number of years in *rwi*. The default value `NULL` lets the function make the decision using some heuristic rules.

`min.corr.overlap`
`numeric` value setting the minimum number of common years in any pair of ring-width series required for their correlation to be included in the calculations. Smaller overlaps are considered to yield unreliable correlation values which are ignored. Defaults to the minimum of 30 and the length of the window. One way to lift the restriction and include all correlations is to set *min.corr.overlap* = `0`.

`round.decimals` non-negative integer `numeric` value setting the desired number of decimal places in the results. Use `NA`, `NULL` or a negative number for no rounding.

`zero.is.missing`
`logical` flag indicating whether to treat zeros as missing values (`TRUE`, the default) or to include them in computation (`FALSE`).

`...` arguments passed on to `rwi.stats.running`

Details

This calculates a variety of descriptive statistics commonly used in dendrochronology.

The function `rwi.stats` is a wrapper that calls `rwi.stats.running` with *running.window* = `FALSE`. The results may differ from those prior to dplR 1.5.3, where the former `rwi.stats` (now renamed to `rwi.stats.legacy`) was replaced with a call to `rwi.stats.running`.

For correctly calculating the statistics on within and between series variability, an appropriate mask (parameter *ids*) must be provided that identifies each series with a tree as it is common for dendrochronologists to take more than one core per tree. The function `read.ids` is helpful for creating a mask based on the series ID.

If *ids* has duplicate tree/core combinations, the corresponding series are averaged before any statistics are computed. The value of the parameter *zero.is.missing* is relevant in the averaging: `TRUE` ensures that zeros don't contribute to the average. The default value of *zero.is.missing* is `TRUE`. The default prior to dplR 1.5.3 was `FALSE`. If the parameter is set to `FALSE`, the user will be warned in case zeros are present. Duplicate tree/core combinations are not detected by `rwi.stats.legacy`.

Row names of *ids* may be used for matching the IDs with series in *rwi*. In this case, the number of rows in *ids* is allowed to exceed the number of series. If some names of *rwi* are missing from the row names of *ids*, the rows of *ids* are assumed to be in the same order as the columns of *rwi*, and the dimensions must match. The latter is also the way that `rwi.stats.legacy` handles *ids*, i.e. names are ignored and dimensions must match.

Note that `period = "common"` can produce `NaN` for many of the stats if there is no common overlap period among the cores. This happens especially in chronologies with floating subfossil samples (e.g., `ca533`).

Some of the statistics are specific to dendrochronology (e.g., the effective number of cores or the expressed population signal). Users unfamiliar with these should see Cook and Kairiukstis (1990) and Fritts (2001) for further details for computational details on the output. The signal-to-noise ratio is calculated following Cook and Pederson (2011).

Note that Buras (2017) cautions against using the expressed population signal as a statistic to determine the whether a chronology correctly represents the population signal of a data set. He reccomends the use of subsample signal strength (`sss`) over EPS.

If desired, the *rwi* can be filtered in the same manner as the family of cross-dating functions using *prewhiten* and *n*. See the help page for `corr.rwl.seg` for more details.

Value

A `data.frame` containing the following columns (each row corresponds to one position of the window):

`start.year`	the first year in the window. Not returned if `running.window` is `FALSE` or called as `rwi.stats` or `rwi.stats.legacy`.
`mid.year`	the middle year in the window, rounded down. Not returned if `running.window` is `FALSE` or called as `rwi.stats` or `rwi.stats.legacy`.
`end.year`	the last year in the window. Not returned if `running.window` is `FALSE` or called as `rwi.stats` or `rwi.stats.legacy`.
`n.cores`	the number of cores
`n.trees`	the number of trees
`n`	the average number of trees (for each year, a tree needs at least one non-`NA` core in order to be counted). Not returned in the results of `rwi.stats.legacy`
`n.tot`	total number of correlations calculated as *n.wt* + *n.bt*. Equal to *n.cores* * (*n.cores*-1)/2 if there is overlap between all samples
`n.wt`	number of within-tree correlations computed
`n.bt`	number of between-tree correlations computed
`rbar.tot`	the mean of all the correlations between different cores
`rbar.wt`	the mean of the correlations between series from the same tree over all trees
`rbar.bt`	the mean interseries correlation between all series from different trees
`c.eff`	the effective number of cores (takes into account the number of within-tree correlations in each tree)
`rbar.eff`	the effective signal calculated as *rbar.bt* / (*rbar.wt* + (1-*rbar.wt*) / *c.eff*)
`eps`	the expressed population signal calculated using the average number of trees as *n* * *rbar.eff* / ((*n* - 1) * *rbar.eff* + 1)
`snr`	the signal to noise ratio calculated using the average number of trees as *n* * *rbar.eff* / (1-*rbar.eff*)

Note

This function uses the foreach looping construct with the %dopar% operator. For parallel computing and a potential speedup, a parallel backend must be registered before running the function.

Author(s)

Mikko Korpela, based on rwi.stats.legacy by Andy Bunn

References

Buras, A. (2017) A comment on the Expressed Population Signal. Dendrochronologia 44:130-132.

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Cook, E. R. and Pederson, N. (2011) Uncertainty, Emergence, and Statistics in Dendrochronology. In Hughes, M. K., Swetnam, T. W., and Diaz, H. F., editors, *Dendroclimatology: Progress and Prospects*, pages 77–112. Springer. ISBN-13: 978-1-4020-4010-8.

Fritts, H. C. (2001) *Tree Rings and Climate*. Blackburn. ISBN-13: 978-1-930665-39-2.

See Also

detrend, cor, read.ids, rwi.stats, corr.rwl.seg

Examples

```
library(utils)
data(gp.rwl)
data(gp.po)
gp.rwi <- cms(rwl = gp.rwl, po = gp.po)
gp.ids <- read.ids(gp.rwl, stc = c(0, 2, 1))
# On a running window
rwi.stats.running(gp.rwi, gp.ids)
## With no running window (i.e. running.window = FALSE)
rwi.stats(gp.rwi, gp.ids)
## Restrict to common overlap (in this case 1899 to 1987)
rwi.stats(gp.rwi, gp.ids, period="common")
rwi.stats.legacy(gp.rwi, gp.ids) # rwi.stats prior to dplR 1.5.3
```

---

rwl.report *Do some reporting on a RWL object*

Description

This function generates a small report on a rwl (or rwi) object that gives the user some basic information on the data including the number of series, the span of the data, the mean interseries correlation, the number of missing rings (zeros), internal NA values, and rings that are very small, or very large.

Usage

```
rwl.report(rwl,small.thresh=NA,big.thresh=NA)
```

Arguments

rwl	a data.frame of ring widths with rownames(x) containing years and colnames(x) containing each series ID such as produced by read.rwl
small.thresh	a numeric value for the threshold value that will cause small rings to be listed. If values is NA this will be omitted.
big.thresh	a numeric value for the threshold value that will cause large rings to be listed. If values is NA this will be omitted.

Details

This generates information about a rwl object including the number of series, the mean length of all the series, the first year, last year, the mean first-order autocorrelation (via summary.rwl), the mean interseries correlation (via interseries.cor), the years where a series has a missing ring (zero), internal NA, very small ring, very large rings, etc.

This output of this function is not typically meant for the user to access but has a print method for the user.

Value

A list with elements containing descriptive information on the rwl object. Specifically:

small.thresh	a numeric value passed in.
big.thresh	a numeric value passed in.
nSeries	a numeric value with the number of series.
n	a numeric value with the total number of rings.
meanSegLength	a numeric with the mean segment length.
firstYear	a numeric with the first year.
lastYear	a numeric with the last year.
meanAR1	a numeric with the mean AR1 value of all series.
sdAR1	a numeric with the standard deviation of the AR1 values of all series.
unconnected	a logical indicating if there are rows (years) with all NA values.
unconnectedYrs	a numeric vector with unconnceted years.
nZeros	a numeric with number of zeros in the data.
zeros	a list containing series and years with zeros or a numeric of 0.
allZeroYears	a numeric indicating rows (years) with all zero values.
consecutiveZeros	a list containing series and years with consecuative zeros or a numeric of 0.
meanInterSeriesCor	a numeric with the mean interseries correlation.

`sdInterSeriesCor`
: a `numeric` with the standard deviation of the interseries correlations.

`internalNAs`
: a `list` containing series and years with internal NA values or a `numeric` of 0.

`smallRings`
: a `list` containing series and years with small rings or a `numeric` of 0.

`bigRings`
: a `list` containing series and years with small rings or a `numeric` of 0.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

`read.rwl`, `summary.rwl`, `interseries.cor`

Examples

```
data("gp.rwl")
rwl.report(rwl = gp.rwl)
# list very small (smallest 1pct) of rings as well
one.pct <- quantile(gp.rwl[gp.rwl != 0], na.rm=TRUE, probs=0.01)
rwl.report(rwl = gp.rwl, small.thresh = one.pct)
```

`rwl.stats` *Calculate Descriptive Summary Statistics on Ring-Width Series*

Description

This function calculates descriptive statistics on a `rwl` object of raw or detrended ring-width series.

Usage

```
rwl.stats(rwl)

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

Arguments

`rwl, object`
: a `rwl` object with (usually) raw ring-width series as columns and years as rows such as that produced by `read.rwl`. It is sometimes desirable to run this on detrended (e.g., rwi) data.

`...`
: Additional arguments from the generic function. These are silently ignored.

Details

This calculates a variety of descriptive statistics commonly used in dendrochronology (see below). Users unfamiliar with these should see Cook and Kairiukstis (1990) and Fritts (2001) for further details.

The summary method for class "rwl" is a wrapper for rwl.stats.

Value

A data.frame containing descriptive stats on each "series". These are the first and last year of the series as well as the length of the series ("first", "last", "year"). The mean, median, standard deviation are given ("mean", "median", "stdev") as are the skewness, the excess kurtosis (calculated as Pearson's kurtosis minus 3), the Gini coefficient, and first order autocorrelation ("skew", "kurtosis", "gini.coef", "ar1").

Note that prior to version 1.6.8, two measures of sensitivity were also included. However mean sensitivity is not a robust statistic that should rarely, if ever, be used (Bunn et al. 2013). Those sensitivity functions ("sens1" and "sens2") are still available for continuity. Users should consider the coef of variation in lieu of mean sensitivity.

Author(s)

Andy Bunn. Slightly improved by Mikko Korpela.

References

Bunn, A. G., Jansma, E., Korpela, M., Westfall, R. D., and Baldwin, J. (2013) Using simulations and data to evaluate mean sensitivity (ζ) as a useful statistic in dendrochronology. *Dendrochronologia*, **31**(3), 250–254.

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Fritts, H. C. (2001) *Tree Rings and Climate*. Blackburn. ISBN-13: 978-1-930665-39-2.

See Also

rwi.stats, read.rwl

Examples

```
library(utils)
data(ca533)
rwl.stats(ca533)
summary(ca533)
```

sea *Superposed Epoch Analysis*

Description

This function calculates the significance of the departure from the mean for a given set of key event years and lagged years.

Usage

```
sea(x, key, lag = 5, resample = 1000)
```

Arguments

x	a chronology data.frame of ring-width indices (such as produced by chron)
key	a vector specifying the key event years for the superposed epoch
lag	an integral value defining the number of lagged years
resample	an integral value specifying the number of bootstrap sample for calculation of confidence intervals

Details

Superposed epoch analysis (SEA) is used to test the significance of a mean tree growth response to certain events (such as droughts). Departures from the mean RWI values for the specified years prior to each event year, the event year, and the specified years immediately after each event are averaged to a superposed epoch. To determine if RWI for these years was significantly different from randomly selected sets of *lag*+1 other years, bootstrap resampling is used to randomly select sets of *lag*+1 years from the data set and to estimate significances for the departures from the mean RWI.

SEA computation is based on scaled RWI values, and 95%-confidence intervals are computed for the scaled values for each year in the superposed epoch.

Value

A data.frame with

lag	the lagged years,
se	the superposed epoch, i.e. the scaled mean RWI for the event years,
se.unscaled	the unscaled superposed epoch, i.e. the mean RWI for the event years,
p	significance of the departure from the chrono's mean RWI,
ci.95.lower	lower 95% confidence band,
ci.95.upper	upper 95% confidence band,
ci.99.lower	lower 99% confidence band,
ci.99.upper	upper 99% confidence band.

Author(s)

Christian Zang. Patched and improved by Mikko Korpela.

References

Lough, J. M. and Fritts, H. C. (1987) An assessment of the possible effects of volcanic eruptions on North American climate using tree-ring data, 1602 to 1900 AD. *Climatic Change*, **10**(3), 219–239.

Examples

```
library(graphics)
library(utils)
data(cana157)
event.years <- c(1631, 1742, 1845)
cana157.sea <- sea(cana157, event.years)
foo <- cana157.sea$se.unscaled
names(foo) <- cana157.sea$lag
barplot(foo, col = ifelse(cana157.sea$p < 0.05, "grey30", "grey75"),
        ylab = "RWI", xlab = "Superposed Epoch")
```

---

seg.plot *Segment Plot*

Description

Makes a segment plot of tree-ring data.

Usage

```
seg.plot(rwl, ...)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
...	arguments to be passed to plot.

Details

This makes a simple plot of the length of each series in a tree-ring data set.

Value

None. This function is invoked for its side effect, which is to produce a plot.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

spag.plot

Examples

```
library(utils)
data(co021)
seg.plot(co021)
```

---

sens1 *Calculate Mean Sensitivity*

Description

This function calculates mean sensitivity of a detrended ring-width series.

Usage

```
sens1(x)
```

Arguments

x	a numeric vector containing the series

Details

This calculates mean sensitivity according to Eq. 1 in Biondi and Qeadan (2008). This is the standard measure of sensitivity in dendrochronology and is typically calculated on detrended series. However, note that mean sensitivity is not a robust statistic and should rarely, if ever, be used (Bunn et al. 2013).

Value

the mean sensitivity.

Author(s)

Mikko Korpela, based on original by Andy Bunn

References

Biondi, F. and Qeadan, F. (2008) Inequality in Paleorecords. *Ecology*, **89**(4), 1056–1067.

Bunn, A. G., Jansma, E., Korpela, M., Westfall, R. D., and Baldwin, J. (2013) Using simulations and data to evaluate mean sensitivity (ζ) as a useful statistic in dendrochronology. *Dendrochronologia*, **31**(3), 250–254.

See Also

sens2, rwl.stats

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
sens1(ca533.rwi[, 1])
```

---

sens2 *Calculate Mean Sensitivity on Series with a Trend*

Description

This function calculates mean sensitivity of a raw or detrended ring-width series.

Usage

```
sens2(x)
```

Arguments

x	a numeric vector containing the series

Details

This calculates mean sensitivity according to Eq. 2 in Biondi and Qeadan (2008). This is a measure of sensitivity in dendrochronology that is typically used in the presence of a trend. However, note that mean sensitivity is not a robust statistic and should rarely, if ever, be used (Bunn et al. 2013).

Value

the mean sensitivity.

Author(s)

Mikko Korpela, based on original by Andy Bunn

References

Biondi, F. and Qeadan, F. (2008) Inequality in Paleorecords. *Ecology*, **89**(4), 1056–1067.

Bunn, A. G., Jansma, E., Korpela, M., Westfall, R. D., and Baldwin, J. (2013) Using simulations and data to evaluate mean sensitivity (ζ) as a useful statistic in dendrochronology. *Dendrochronologia*, **31**(3), 250–254.

See Also

sens1, rwl.stats

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
sens2(ca533.rwi[, 1])
```

---

series.rwl.plot *Plot Series and a Master*

Description

Plots a tree-ring series with a master chronology and displays their fit, segments, and detrending options in support of the cross-dating functions.

Usage

```
series.rwl.plot(rwl, series, series.yrs = as.numeric(names(series)),
                seg.length = 100, bin.floor = 100, n = NULL,
                prewhiten = TRUE, biweight = TRUE, floor.plus1 = FALSE)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)). Ignored if *series* is an index to a column of *rwl*.
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
bin.floor	a non-negative integral value giving the base for locating the first segment (e.g., 1600, 1700, 1800 AD). Typically 0, 10, 50, 100, etc.
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
prewhiten	logical flag. If TRUE each series is whitened using ar.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).

Details

The function is typically invoked to produce four plots showing the effect of the detrending options *n* and *prewhiten* and the binning options *seg.length* and *bin.floor*.

Plot 1 Time series plot of the filtered series and the master

Plot 2 Scatterplot of series vs. master

Plot 3 Segments that would be used in the other cross-dating functions (e.g., `corr.series.seg`)

Plot 4 Text giving the detrending options and the time span of the raw and filtered series and master

The series and master are returned as well.

See help pages for `corr.rwl.seg`, `corr.series.seg`, and `ccf.series.rwl` for more information on these arguments.

Value

A `list` containing the filtered vectors *series* and *master*.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

`corr.rwl.seg`, `corr.series.seg`, `ccf.series.rwl`

Examples

```
library(utils)
data(co021)
foo <- series.rwl.plot(rwl = co021, series = "646244", seg.length = 100,
                       n = 5)
## note effect of n on first year in the series
foo <- series.rwl.plot(rwl = co021, series = "646244", seg.length = 100,
                       n = 13, prewhiten = FALSE)
bar <- series.rwl.plot(rwl = co021, series = "646244", seg.length = 100,
                       n = 7, prewhiten = FALSE)
head(foo$series)
head(bar$series)
```

sgc *Synchronous Growth Changes*

Description

This function calculates the synchronous growth changes (sgc), semi synchronous growth changes (ssgc) and the length of the compared overlap for a given set of tree-ring records. Optionally the probability of exceedence is calculated.

Usage

```
sgc(x,overlap = 50, prob = TRUE)
```

Arguments

x	a data.frame of tree-ring data with records in columns, and years as rows.
overlap	integer value with minimal length of overlapping growth changes (compared number of tree rings - 1). Comparisons with less overlap are not compared.
prob	if TRUE then the probability of exceedence of the sgc will be calculated

Details

The sgc is a non parametric test based on sign tests.The synchronous growth changes (sgc) and semi synchronous growth changes (ssgc) are meant to replace the Gleichläufigkeit (glk()), since the Gleichläufigkeit can be (strongly) influenced by years when one of the compared series shows no growth change. The sgc gives a better description of the similarity (Visser, 2020). The ssgc gives the percentage years that one of the compared series shows no growth change. This function implements sgc and ssgc as the vectorized pairwise comparison of all records in data set.

The probability of exceedence (p) for the sgc expresses the chance that the sgc is incorrect. The observed value of the sgc is converted to a z-score and based on the standard normal curve the probability of exceedence is calculated (Visser 2020). The result is a matrix of all p-values.

Value

A list with three or four matrices (p_mat is optional if prob = TRUE):

1. sgc_mat: matrix with synchronous growth changes (sgc) for all possible combinations of records
2. ssgc_mat: matrix with semi-synchronous growth changes (ssgc) for all possible combinations of records
3. overlap: matrix with number of overlapping growth changes.This is the number of overlapping years minus one.
4. p_mat: matrix of all probabilities of exceedence for all observed sgc values.

The matrices can be extracted from the list by selecting the name or the index number. Comparisons are only compared if the overlap is above the set theshold and if no threshold is set, this defaults to 50 years.If no comparison can be compared, NA is returned.

To calculate the global sgc of the dataset (assuming `x.sgc <- sgc(x)`: `mean(x.sgc$sgc_mat, na.rm = TRUE))`. For the global ssgc use: `mean(x.sgc$ssgc_mat, na.rm = TRUE)`.

Author(s)

Ronald Visser

References

Visser, R.M. (2020) On the similarity of tree-ring patterns: Assessing the influence of semi-synchronous growth changes on the Gleichläufigkeit for big tree-ring data sets,*Archaeometry*, **63**, 204-215 DOI: https://doi.org/10.1111/arcm.12600

See Also

glk

Examples

```
library(dplR)
data(ca533)
ca533.sgclist <- sgc(ca533)
mean(ca533.sgclist$sgc_mat, na.rm = TRUE)
mean(ca533.sgclist$ssgc_mat, na.rm = TRUE)
```

---

skel.plot *Skeleton Plot*

Description

Automatically generates a skeleton plot of tree-ring data.

Usage

```
skel.plot(rw.vec, yr.vec = NULL, sname = "", filt.weight = 9,
          dat.out = FALSE, master = FALSE, plot = TRUE)
```

Arguments

`rw.vec`	a `numeric` vector of a tree-ring chronology or series
`yr.vec`	optional `numeric` vector giving years for the tree-ring data
`sname`	an optional `character` string giving the ID for the data. The width of the string, often identical to the number of characters, must be less than 8.
`filt.weight`	filter length for the Hanning filter, defaults to 9

dat.out	logical flag indicating whether to return a data.frame containing the data.
master	logical flag indicating whether to make the plot with the segments inverted to ease pattern matching. If TRUE the segments will be plotted from the top down and the labels on the x axes will be on the bottom.
plot	logical flag indicating whether to make a plot.

Details

This makes a skeleton plot – a plot that gives the relative growth for year t relative to years t-1 and t+1. Note that this plot is a standard plot in dendrochronology and typically made by hand for visually cross-dating series. This type of plot might be confusing to those not accustomed to visual cross-dating. See references for more information. The implementation is based on Meko's (2002) skeleton plotting approach.

The skeleton plot is made by calculating departures from high frequency growth for each year by comparing year t to the surrounding three years (t-1,t,t+1). Low frequency variation is removed using a hanning filter. Relative growth is scaled from one to ten but only values greater than three are plotted. This function's primary effect is to create plot with absolute units that can be printed and compared to other plots. Here, anomalous growth is plotted on a 2mm grid and up to 120 years are plotted on a single row with a maximum of 7 rows (840 years). These plots are designed to be plotted on standard paper using an appropriate device, e.g., postscript with defaults or to pdf with plot width and height to accommodate a landscape plot, e.g., *width* = 10, *height* = 7.5, *paper* = "USr". These plots are designed to be printable and cut into strips to align long series. Statistical cross-dating is possible if the data are output but more easily done using the functions xskel.plot and xskel.ccf.plot.

Value

This function is invoked primarily for its side effect, which is to produce a plot. If *dat.out* is TRUE then a data.frame is returned with the years and height of the skeleton plot segments as columns.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Stokes, M. A. and Smiley, T. L. (1968) *An Introduction to Tree-Ring Dating*. The University of Arizona Press. ISBN-13: 978-0-8165-1680-3.

Sheppard, P. R. (2002) Crossdating Tree Rings Using Skeleton Plotting. https://www.ltrr.arizona.edu/skeletonplot/introcrossdate.htm.

Meko, D. (2002) Tree-Ring MATLAB Toolbox. https://www.mathworks.com/matlabcentral/.

See Also

Devices, hanning, xskel.plot, xskel.ccf.plot

Examples

```
library(utils)
data(co021)
x <- co021[,33]
x.yrs <- time(co021)
x.name <- colnames(co021)[33]
## On a raw ring width series - undated
skel.plot(x)
## On a raw ring width series - dated with names
skel.plot(x, yr.vec = x.yrs, sname = x.name, master = TRUE)
## Not run:
library(grDevices)
## Try cross-dating
y <- co021[, 11]
y.yrs <- time(co021)
y.name <- colnames(co021)[11]

## send to postscript - 3 pages total
fname1 <- tempfile(fileext=".ps")
print(fname1) # tempfile used for PS output
postscript(fname1)
## "Master series" with correct calendar dates
skel.plot(x, yr.vec = x.yrs, sname = x.name, master = TRUE)
## Undated series, try to align with last plot
skel.plot(y)
## Here's the answer...
skel.plot(y, yr.vec = y.yrs, sname = y.name)
dev.off()

unlink(fname1) # remove the PS file

## alternatively send to pdf
fname2 <- tempfile(fileext=".pdf")
print(fname2) # tempfile used for PDF output
pdf(fname2, width = 10, height = 7.5, paper = "USr")
skel.plot(x, yr.vec = x.yrs, sname = x.name, master = TRUE)
skel.plot(y)
skel.plot(y, yr.vec = y.yrs, sname = y.name)
dev.off()

unlink(fname2) # remove the PDF file

## End(Not run)
```

---

spag.plot *Spaghetti Plot*

Description

Makes a spaghetti plot of tree-ring data.

Usage

```
spag.plot(rwl, zfac = 1, useRaster = FALSE, res = 150, ...)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
zfac	a multiplier for *rwl* to enhance clarity when plotting.
useRaster	A logical flag. If TRUE, the tree-ring series are drawn as a raster image. Other parts of the plot are not affected. Other choices are NA (automatic) and FALSE (use vector graphics, the default). See wavelet.plot.
res	A numeric vector of length 1. The resolution (pixels per inch) of the tree-ring series when a raster image is used.
...	arguments to be passed to lines.

Details

This makes a simple plot of each series in a tree-ring data set. Each series is centered first by subtracting the column mean using scale. The plot can be grossly tuned with *zfac* which is a multiplier to *rwl* before plotting and centering.

Value

None. This function is invoked for its side effect, which is to produce a plot.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

seg.plot

Examples

```
library(utils)
data(co021)
plot(co021,plot.type = "spag")
spag.plot(co021, zfac = 2)
```

ssf *Simple Signal Free Standardization*

Description

A simple implentation of the signal-free chronology

Usage

```
ssf(rwl,
    method="Spline",
    nyrs = NULL,
    difference = FALSE,
    max.iterations = 25,
    mad.threshold = 5e-4,
    recode.zeros = FALSE,
    return.info = FALSE,
    verbose = TRUE)
```

Arguments

`rwl`	a `rwl` object with series as columns and years as rows such as that produced by `read.rwl`. See 'Details'.
`method`	a `character` vector to determine the detrending method. See 'Details' below. Possible values are `c("Spline", "AgeDepSpline")`.
`nyrs`	a number controlling the smoothness of the fitted curve in methods. See 'Details' in `detrend.series`.
`difference`	a `logical` flag. Compute residuals by subtraction if TRUE, otherwise use division. See 'Details' in `detrend.series`
`max.iterations`	a `numeric`. The maximum number of iterations to be used in the fitting process. See 'Details'.
`mad.threshold`	a `numeric`. The median absolute difference threshold used for a stopping criteria. See 'Details'.
`recode.zeros`	a `logical` flag. Replace values of zero in `rwl` with 0.001 if TRUE.
`return.info`	a `logical` flag. If `TRUE`, details about models and data will be added to the return value. See 'Value'.
`verbose`	a `logical` flag. Write out details to the screen?

Details

This function creates a simple signal-free chronology that loosely follows the procedures laid out on p75 of Melvin and Briffa (2008). This function is a lighter version of that procedure and users who want more control and refinement should look to the CRUST program decried in Melvin and Briffa (2014). These steps are described in more detail in Learning to Love R.

1. Detrend each series using the selected method, calculate RWI by division (or subtraction), and create an initial mean-value chronology.
2. Create signal-free measurements by dividing (or subtracting) each series of measurements by the chronology. If `return.info` is invoked these are returned in `sfRW_Array`.
3. Rescale the signal-free measurements to their original mean. If `return.info` is invoked these are returned in `sfRWRescaled_Array`.
4. If the sample depth is one, replace signal-free measurements with original measurements.
5. Fit curves to signal free measurements.If `return.info` is invoked these are returned in `sfRWRescaledCurves_Array`.
6. Get new growth indicies by dividing (or subtracting) the original measurements by curves in the last step. If `return.info` is invoked these are returned in `sfRWI_Array`.
7. Create a mean-value chronology using the indicies from the prior step. If `return.info` is invoked these are returned in `sfCrn_Mat`.
8. Repeat steps two through seven up to `maxIter` or until the `madThreshold` is reached. The stopping criteria is determined using the absolute difference between filtered chronologies generated in interation k and k-1. This is done with the residuals of a high-pass filter on the chronology using a cubic smoothing spline (`caps`) with the stiffness set as the median of the segment lengths of series contributing to the chronology. The stopping threshold is calculated as the median absolute difference of the kth and kth-1 chronologies weighted by the normalized sample depth. If `return.info` is invoked the residual chronologies are returned in `hfCrnResids_Mat` and the median absolute differences are returns in `MAD_Vec`.

The input object (`rwl`) should be of `class` `rwl`. If it not, the function will attempt to coerce it using `as.rwl` and a warning will be issued.

See the references below for further details on detrending. It's a dark art.

Value

An object of of class `crn` and `data.frame` with the signal-free chronology and the sample depth. The years are stored as row numbers.

If `return.info` is `TRUE` a `list` containing ouptput by iteration (k):

`infoList` a `list` with information on the arguments used in the function call.

`k` a `numeric` containing the number of iterations.

`ssfCrn` the signal-free chronology as above.

`sfRW_Array` an `array` of years by series by iteration (0:k) that holds the signal free measurements. The values in `sfRW_Array[,,1]` are what is used at the start of the iterative process (iteration 0) which here are the original data.

`sfRWRescaled_Array` an `array` of years by series by iteration (0:k) that holds the rescaled signal free measurements. The values in `sfRWRescaled_Array[,,1]` are what is used at the start of the iterative process (iteration 0) which here are the original data.

`sfRWRescaledCurves_Array` an `array` of years by series by iteration (0:k) that holds the rescaled signal free curve fits. The values in `sfRWRescaledCurves_Array[,,1]` are the curves used at the start of the iterative process (iteration 0).

`sfRWI_Array` an `array` of years by series by iteration (0:k) that holds the detrended signal free measurements. The values in `sfRWI_Array[,,1]` are the RWI values at the start of the iterative process (iteration 0).

`sfCrn_Mat` a `matrix` of years by iteration (0:k) that holds the calculated chronology. The values in `sfCrn_Mat[,1]` is the initial chronology from the start of the iterative process (iteration 0).

`hfCrn_Mat` a `matrix` of years by iteration (0:k) that holds the calculated high-frequency chronology. The values in `hfCrn_Mat[,1]` is the initial high-frequency chronology from the start of the iterative process (iteration 0).

`hfCrnResids_Mat` a `matrix` of years by k-1 that holds the differences between the kth and the kth-1 high frequency chronology residuals.

`MAD_out` a `vector` containing the median absolute difference between iteration k and k-1.

Author(s)

Ed Cook provided Fortran code that was ported to R by Andy Bunn.

References

Melvin, TM, Briffa, KR (2008) A 'signal-free' approach to dendroclimatic standardisation. Dendrochronologia 26: 71–86 doi: 10.1016/j.dendro.2007.12.001

Melvin T. M. and Briffa K.R. (2014a) CRUST: Software for the implementation of Regional Chronology Standardisation: Part 1. Signal-Free RCS. Dendrochronologia 32, 7-20, doi: 10.1016/j.dendro.2013.06.002

Melvin T. M. and Briffa K.R. (2014b) CRUST: Software for the implementation of Regional Chronology Standardisation: Part 2. Further RCS options and recommendations. Dendrochronologia 32, 343-356, doi: 10.1016/j.dendro.2014.07.008

See Also

`detrend`. `chron`

Examples

```
library(stats)
data(wa082)
wa082_SSF <- ssf(wa082)
plot(wa082_SSF,add.spline=TRUE,nyrs=20)

wa082_SSF_Full <- ssf(wa082,method = "AgeDepSpline",
  difference = TRUE, return.info = TRUE)
plot(wa082_SSF_Full$ssfCrn,add.spline=TRUE,nyrs=20)
```

sss *Subsample Signal Strength*

Description

Calculate subsample signal strength on a `data.frame` of (usually) ring-width indices.

Usage

```
sss(rwi, ids = NULL)
```

Arguments

`rwi`	a `data.frame` with detrended and standardized ring width indices as columns and years as rows such as that produced by `detrend`.
`ids`	an optional `data.frame` with column one named `"tree"` giving a `numeric` ID for each tree and column two named `"core"` giving a `numeric` ID for each core. Defaults to one core per tree as `data.frame(tree=1:ncol(rwi), core=rep(1, ncol(rwi)))`.

Details

This calculates subsample signal strength (sss) following equation 3.50 in Cook and Kairiukstis (1990) but using notation from Buras (2017) because writing the prime unicode symbol seems too difficult. The function calls `rwi.stats` and passes it the arguments `ids` and `prewhiten`.

To make better use of variation in growth within and between series, an appropriate mask (parameter *ids*) should be provided that identifies each series with a tree as it is common for dendrochronologists to take more than one core per tree. The function `read.ids` is helpful for creating a mask based on the series ID.

Subsample signal strength is calculated as $\frac{n[1+(N-1)\bar{r}]}{N[1+(n-1)\bar{r}]}$ where `n` and `N` are the number of cores or trees in the subsample and sample respectively and `rbar` is mean interseries correlation. If there is only one core per tree `n` is the sample depth in a given year (`rowSums(!is.na(rwi))`), `N` is the number of cores (`n.cores` as given by `rwi.stats`), and `rbar` is the mean interseries correlation between all series (`r.bt` as given by `rwi.stats`). If there are multiple cores per tree `n` is the number of trees present in a given year, `N` is the number of trees (`n.trees` as given by `rwi.stats`), and `rbar` is the effective mean interseries correlation (`r.eff` as given by `rwi.stats`).

Readers interested in the differences between subsample signal strength and the more commonly used (running) expressed population signal should look at Buras (2017) on the common misuse of the expressed population signal as well as Cook and Pederson (2011) for a more general approach to categorizing variability in tree-ring data.

Value

A `numeric` containing the subsample signal strength that is the same as number if rows of`rwi`.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Buras, A. (2017) A comment on the Expressed Population Signal. Dendrochronologia 44:130-132.

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0-7923-0586-6.

Cook, E. R. and Pederson, N. (2011) Uncertainty, Emergence, and Statistics in Dendrochronology. In Hughes, M. K., Swetnam, T. W., and Diaz, H. F., editors, *Dendroclimatology: Progress and Prospects*, pages 77–112. Springer. ISBN-13: 978-1-4020-4010-8.

See Also

rwi.stats, read.ids

Examples

```
data(ca533)
ca533.rwi <- detrend(ca533,method="Spline")
# assuming 1 core / tree
ca533.sss <- sss(ca533.rwi)

ca533.ids <- autoread.ids(ca533)
# done properly with >=1 core / tree as per the ids
ca533.sss2 <- sss(ca533.rwi,ca533.ids)

yr <- time(ca533)
plot(yr,ca533.sss,type="l",ylim=c(0.4,1),
     col="darkblue",lwd=2,xlab="Year",ylab="SSS")
lines(yr,ca533.sss2,lty="dashed",
      col="darkgreen",lwd=2)

# Plot the chronology showing a potential cutoff year based on SSS
# (using sss2 with the correct series IDs to get >=1 core / tree as per the ids)
ca533.crn <- chron(ca533.rwi)
def.par <- par(no.readonly=TRUE)
par(mar = c(2, 2, 2, 2), mgp = c(1.1, 0.1, 0), tcl = 0.25, xaxs='i')
plot(yr, ca533.crn[, 1], type = "n", xlab = "Year",
     ylab = "RWI", axes=FALSE)
cutoff <- max(yr[ca533.sss2 < 0.85])
xx <- c(500, 500, cutoff, cutoff)
yy <- c(-1, 3, 3, -1)
polygon(xx, yy, col = "grey80")
abline(h = 1, lwd = 1.5)
lines(yr, ca533.crn[, 1], col = "grey50")
lines(yr, caps(ca533.crn[, 1], nyrs = 32), col = "red", lwd = 2)
axis(1); axis(2); axis(3);
par(new = TRUE)
## Add SSS
plot(yr, ca533.sss2, type = "l", xlab = "", ylab = "",
```

```
     axes = FALSE, col = "blue")
abline(h=0.85,col="blue",lty="dashed")
axis(4, at = pretty(ca533.sss2))
mtext("SSS", side = 4, line = 1.1, lwd=1.5)
box()
par(def.par)
```

`strip.rwl` *Chronology Stripping by* EPS

Description

EPS-based chronology stripping after Fowler & Boswijk 2003.

Usage

```
strip.rwl(rwl, ids = NULL, verbose = FALSE, comp.plot = FALSE,
          legacy.eps = FALSE)
```

Arguments

`rwl`	a `data.frame` of raw tree-ring widths series, such as that produced by `read.rwl` or `read.fh`
`ids`	an optional `data.frame` with column one named `"tree"` giving a `numeric` ID for each tree and column two named `"core"` giving a `numeric` ID for each core. This is passed on to `rwi.stats`. See its manual for the meaning of the default value `NULL` and more information.
`verbose`	`logical` flag, indicating if the EPS calculated at each step and other details should be printed on the console during the chronology stripping process
`comp.plot`	`logical` flag, indicating if a diagnostic plot with year-wise stripped and unstripped EPS should be drawn (see details below)
`legacy.eps`	`logical` flag, indicating if the EPS will be calculated with `rwi.stats` (`FALSE`, the default) or `rwi.stats.legacy` (`TRUE`)

Details

The EPS-based chronology stripping is implemented after Fowler & Boswijk 2003: First, all series are standardized using a double detrending procedure with splines and frequency cutoffs of 50% at 20 and 200 years. Then, EPS is calculated for the chronology including all (remaining) series. In each iteration, the algorithm calculates leave-one-out EPS values, and the series whose removal increases overall EPS the most is discarded. This is repeated until no further increase in EPS is gained by discarding a single series. The procedure is then repeated in the opposite direction, i.e., the reinsertion of each previously removed series into the `data.frame` is considered. In each iteration, the series (if any) whose reinsertion increases EPS the most is reinserted. As a last step, EPS is calculated for each year of the stripped and original chronology including all series. If `comp.plot` is set to `TRUE`, a diagnostic plot is shown for the year-wise comparison.

When verbose output is chosen, the EPS values for all leave-one-out (or back-in) chronologies are reported. If discarding or re-inserting a single series leads to an improvement in EPS, this series is marked with an asterisk.

Value

The functions returns a `data.frame` of raw tree-ring widths, where series that do not contribute to an overall improvement in EPS are left out.

Author(s)

Christian Zang. Patched and improved by Mikko Korpela.

References

Fowler, A. and Boswijk, G. (2003) Chronology stripping as a tool for enhancing the statistical quality of tree-ring chronologies. *Tree-Ring Research*, **59**(2), 53–62.

See Also

`rwi.stats`

Examples

```
library(utils)
data(anos1)
anos1.ids <- read.ids(anos1, stc = c(4, 3, 1))
srwl <- strip.rwl(anos1, ids = anos1.ids, verbose = TRUE)
tail(srwl)
```

---

tbrm *Calculate Tukey's Biweight Robust Mean*

Description

This calculates a robust average that is unaffected by outliers.

Usage

```
tbrm(x, C = 9)
```

Arguments

`x`	a `numeric` vector
`C`	a constant. C is preassigned a value of 9 according to the Cook reference below but other values are possible.

Details

This is a one step computation that follows the Affy whitepaper below, see page 22. This function is called by chron to calculate a robust mean. C determines the point at which outliers are given a weight of 0 and therefore do not contribute to the calculation of the mean. C = 9 sets values roughly +/-6 standard deviations to 0. C = 6 is also used in tree-ring chronology development. Cook and Kairiukstis (1990) have further details.

An exact summation algorithm (Shewchuk 1997) is used. When some assumptions about the rounding of floating point numbers and conservative compiler optimizations hold, summation error is completely avoided. Whether the assumptions hold depends on the platform, i.e. compiler and CPU.

Value

A numeric mean.

Author(s)

Mikko Korpela

References

Statistical Algorithms Description Document, 2002, Affymetrix.

Cook, E. R. and Kairiukstis, L. A., editors (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences.* Springer. ISBN-13: 978-0-7923-0586-6.

Mosteller, F. and Tukey, J. W. (1977) *Data Analysis and Regression: a second course in statistics*. Addison-Wesley. ISBN-13: 978-0-201-04854-4.

Shewchuk, J. R. (1997) Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete and Computational Geometry*, **18**(3), 305–363.

See Also

chron

Examples

```
library(stats)
library(utils)
foo <- rnorm(100)
tbrm(foo)
mean(foo)

## Compare
data(co021)
co021.rwi <- detrend(co021, method = "ModNegExp")
crn1 <- apply(co021.rwi, 1, tbrm)
crn2 <- chron(co021.rwi)
cor(crn1, crn2[, 1])
```

time.rwl *Retrieve or set the time values for rwl and crn objects*

Description

Retrieve or set the time values for rwl and crn objects.

Usage

```
## S3 method for class 'rwl'
time(x, ...)
## S3 method for class 'crn'
time(x, ...)
time(x) <- value
```

Arguments

x	An object of class "rwl" or an aboject of class "crn".
...	Not used.
value	A numeric vector to assign the time.

Value

A numeric vector of time (typically in years) for the object. This is done via as.numeric(rownames(x)) but has been asked for by users so many times that it is being included as a function.

Author(s)

Andy Bunn

See Also

read.rwl read.crn

Examples

```
library(utils)
data(co021)
# extract years
co021.yrs <- time(co021)
# set years -- silly example
time(co021) <- co021.yrs+100
```

treeMean *Calculate mean across cores in a tree*

Description

This function calculates the mean value for each tree in a rwl or rwi object.

Usage

```
treeMean(rwl, ids, na.rm=FALSE)
```

Arguments

rwl	a data.frame of ring widths with rownames(*rwl*) containing years and colnames(rwl) containing each series ID such as produced by read.rwl
ids	a data.frame with column one named "tree" giving a numeric ID for each tree and column two named "core" giving a numeric ID for each core.
na.rm	logical passed to rowMeans. Should missing values be removed?

Details

This function averages together multiple cores to give a mean value of growth. It is very common in dendrochronology to take more than one core per tree. In those cases it is occassionally desirable to have an average of the cores. This function merely loops through the rwl object and calculates the rowMeans for each tree. If na.rm=TRUE trees with >1 sample will be averaged only over the period where the samples overlap. If FALSE the output can vary in the number of samples. See examples.

Value

An object of class c("rwl", "data.frame") with the mean annual value for each tree.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

read.rwl, read.ids

Examples

```
data(gp.rwl)
gp.ids <- read.ids(gp.rwl, stc = c(0, 2, 1))

gp.treeMean <- treeMean(gp.rwl, gp.ids)
gp.treeMean2 <- treeMean(gp.rwl, gp.ids, na.rm=TRUE)

# look at an example of a single tree with different averaging periods
```

```
tree40 <- data.frame(gp.rwl[, c("40A","40B")],
                     gp.treeMean[, "40", drop=FALSE],
                     gp.treeMean2[, "40", drop=FALSE])
names(tree40) <- c("coreA", "coreB", "treeMean1", "treeMean2")
head(tree40,50)

data(ca533)
ca533.treeMean <- treeMean(ca533, autoread.ids(ca533))
# plot using S3method for class "rwl"
plot(ca533.treeMean,plot.type="spag")
```

`tridas.vocabulary` *Browse and Check Standard TRiDaS Vocabulary*

Description

This function can be used to browse the TRiDaS vocabulary by category.

Usage

```
tridas.vocabulary(category = c("dating type", "measuring method",
                      "shape", "location type", "variable", "unit",
                      "remark", "dating suffix", "presence / absence",
                      "complex presence / absence", "certainty"),
                    idx = NA, term = NA, match.exact = FALSE)
```

Arguments

`category`	Vocabulary category as a `character` vector of length one. One of `"dating type"`, `"measuring method"`, `"shape"`, `"location type"`, `"variable"`, `"unit"`, `"remark"`, `"dating suffix"`, `"presence / absence"`, `"complex presence / absence"`, `"certainty"`. Partial matches are allowed.
`idx`	A `numeric` vector. Index to the `character` vector containing the vocabulary of the given category.
`term`	A `character` vector. One or more (partial) terms to look for in the given category.
`match.exact`	A `logical` value. If `TRUE`, partial matching of *term* is not used. Defaults to `FALSE`.

Details

The Tree Ring Data Standard (TRiDaS) is described in Jansma et. al (2010).

The function has four usage modes:

1. When *idx* is given, returns item number *idx* in the given *category*. There may be several numbers in *idx*, in which case multiple items are returned.

2. When *term* contains one or more items and *match.exact* is `TRUE`, checks whether any of the terms is an exact match in the given *category*
3. When *term* contains one or more items and *match.exact* is `FALSE`, expands partial matches of the terms in the vocabulary of the given *category*
4. When only *category* is given, returns the complete vocabulary in the given *category*

Value

In mode 1	A `character` vector, same length as in *idx*
In mode 2	A `logical` value
In mode 3	A `character` vector, same length as in *term*
In mode 4	A `character` vector

Author(s)

Mikko Korpela

References

Jansma, E., Brewer, P. W., and Zandhuis, I. (2010) TRiDaS 1.1: The tree-ring data standard. *Dendrochronologia*, **28**(2), 99–130.

See Also

`read.tridas`, `write.tridas`

Examples

```
## Show all entries in category "measuring method"
tridas.vocabulary(category = "measuring")

## Show item number one in category "complex presence / absence"
tridas.vocabulary(category = "complex", idx = 1)

## Check whether "half section" exists in category "shape"
tridas.vocabulary(category = "shape", term = "half section",
                  match.exact = TRUE)

## Return unabbreviated matches to several queries in category "remark"
tridas.vocabulary(category = "remark",
                  term = c("trauma", "fire", "diffuse"))
```

`uuid.gen` UUID *Generator*

Description

Initializes and returns a generator of universally unique identifiers. Use the returned function repeatedly for creating one or more UUIDs, one per function call.

Usage

```
uuid.gen(more.state = "")
```

Arguments

`more.state`	A `character` string for altering the state of the generator

Details

This function returns a function (closure) which generates UUIDs. The state of that anonymous function is set when `uuid.gen` is called. The state consists of the following:

- System and user information (`Sys.info`)
- R version (`R.version`)
- Platform information (`.Platform`)
- Working directory
- Process ID of the R session
- Time when `uuid.gen` was called (precision of seconds or finer)
- The text in parameter *more.state*

The Pseudo Random Number Generator of R (see `.Random.seed`) is used in the generation of UUIDs. No initialization of the PRNG is done. Tampering with the state of the R PRNG while using a given UUID generator causes a risk of non-unique identifiers. Particularly, setting the state of the PRNG to the same value before two calls to the UUID generator guarantees two identical identifiers. If two UUID generators have a different state, it is *not* a problem to have the PRNG going through or starting from the same state with both generators.

The user is responsible for selecting a PRNG with a reasonable number of randomness. Usually, this doesn't require any action. For example, any PRNG algorithm available in R works fine. However, the uniqueness of UUIDs can be destroyed by using a bad user-supplied PRNG.

The UUIDs produced by `uuid.gen` generators are Version 4 (random) with 122 random bits and 6 fixed bits. The UUID is presented as a `character` string of 32 hexadecimal digits and 4 hyphens:

'`xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`'

where *x* is any hexadecimal digit and *y* is one of `"8"`, `"9"`, `"a"`, or `"b"`. Each *x* and *y* in the example is an independent variables (for all practical purposes); subscripts are omitted for clarity. The UUID generator gets 32 hex digits from the MD5 message digest algorithm by feeding it a string consisting of the constant generator state and 5 (pseudo) random numbers. After that, the 6 bits are fixed and the hyphens are added to form the final UUID.

Value

A parameterless function which returns a single UUID (character string)

Author(s)

Mikko Korpela

References

Leach, P., Mealling, M., and Salz, R. (2005) A Universally Unique IDentifier (UUID) URN namespace. RFC 4122, RFC Editor. https://www.rfc-editor.org/rfc/rfc4122.txt.

See Also

digest, Random

Examples

```
## Normal use
ug <- uuid.gen()
uuids <- character(100)
for(i in 1:100){
  uuids[i] <- ug()
}
length(unique(uuids)) == 100 # TRUE, UUIDs are unique with high probability

## Do NOT do the following unless you want non-unique IDs
rs <- .Random.seed
set.seed(0L)
id1 <- ug()
set.seed(0L)
id2 <- ug()
id1 != id2 # FALSE, The UUIDs are the same
.Random.seed <- rs

## Strange usage pattern, but will probably produce unique IDs
ug1 <- uuid.gen("1")
set.seed(0L)
id1 <- ug1()
ug2 <- uuid.gen("2")
set.seed(0L)
id2 <- ug2()
id1 != id2 # TRUE, The UUIDs are different with high probability
.Random.seed <- rs
```

wa082 *Hurricane Ridge, Pacific silver fir*

Description

This data set gives the raw ring widths for Pacific silver fir *Abies amabilis* at Hurricane Ridge in Washington, USA. There are 23 series. Data set was created using `read.rwl` and saved to an .rda file using `save`.

Usage

```
data(wa082)
```

Format

A `data.frame` containing 23 tree-ring series in columns and 286 years in rows.

Source

International tree-ring data bank, Accessed on 20-April-2021 at https://www.ncei.noaa.gov/pub/data/paleo/treering/measurements/northamerica/usa/wa082.rwl

References

Schweingruber, F. (1983) Hurricane Ridge Data Set. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series 1983-wa082.RWL. NOAA/NCDC Paleoclimatology Program, Boulder, Colorado, USA.

wavelet.plot *Plot a Continuous Wavelet Transform*

Description

This function creates a `filled.contour` plot of a continuous wavelet transform as output from `morlet`.

Usage

```
wavelet.plot(wave.list,
             wavelet.levels = quantile(wave.list$Power,
                                       probs = (0:10)/10),
             add.coi = TRUE, add.sig = TRUE,
             x.lab = gettext("Time", domain = "R-dplR"),
             period.lab = gettext("Period", domain = "R-dplR"),
             crn.lab = gettext("RWI", domain = "R-dplR"),
             key.cols = rev(rainbow(length(wavelet.levels)-1)),
```

```
                key.lab = parse(text=paste0("\"",
                                            gettext("Power",
                                                    domain="R-dplR"),
                                            "\"^2")),
                add.spline = FALSE, f = 0.5, nyrs = NULL,
                crn.col = "black", crn.lwd = 1,coi.col='black',
                crn.ylim = range(wave.list$y) * c(0.95, 1.05),
                side.by.side = FALSE,
                useRaster = FALSE, res = 150, reverse.y = FALSE, ...)
```

Arguments

wave.list	A list. Output from morlet.
wavelet.levels	A numeric vector. Values for levels of the filled contours for the wavelet plot.
add.coi	A logical flag. Add cone of influence?
add.sig	A logical flag. Add contour lines for significance?
x.lab	X-axis label.
period.lab	Y-axis label for the wavelet plot.
crn.lab	Y-axis label for the time-series plot.
key.cols	A vector of colors for the wavelets and the key.
key.lab	Label for key.
add.spline	A logical flag. Add a spline to the time-series plot using caps?.
nyrs	A number giving the rigidity of the smoothing spline, defaults to 0.33 of series length if *nyrs* is NULL.
f	A number between 0 and 1 giving the frequency response or wavelength cutoff for the smoothing spline. Defaults to 0.5.
crn.col	Line color for the time-series plot.
crn.lwd	Line width for the time-series plot.
coi.col	Color for the COI if *add.coi* is TRUE.
crn.ylim	Axis limits for the time-series plot.
side.by.side	A logical flag. Plots will be in one row if TRUE.
useRaster	A logical flag. If TRUE, the filled contours are drawn as a raster image. Other parts of the plot are not affected. useRaster=TRUE can be especially useful when a pdf device is used: the size and complexity of the PDF file will probably be greatly reduced. Setting this to TRUE has negative effects when used with a bitmap device such as png. If NA, plotting of a raster image will be attempted if and only if the name of the graphics device is "pdf" or "postscript". The default is FALSE: draw directly to the graphics device without using an intermediate raster image.
res	A numeric vector of length 1. The resolution (pixels per inch) of the filled contours when a raster image is used. See *useRaster*.
reverse.y	A logical flag. If TRUE, the Y-axis will be reversed, i.e. period increasing towards the bottom. The default is FALSE.
...	Arguments passed to rasterPlot. Only relevant when the filled contours are drawn as a raster image. See *useRaster*.

Details

This produces a plot of a continuous wavelet transform and plots the original time series. Contours are added for significance and a cone of influence polygon can be added as well. Anything within the cone of influence should not be interpreted.

The time series can be plotted with a smoothing spline as well.

Value

None. This function is invoked for its side effect, which is to produce a plot.

Note

The function `morlet` is a port of Torrence's IDL code, which can be accessed through the Internet Archive Wayback Machine.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Torrence, C. and Compo, G. P. (1998) A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, **79**(1), 61–78.

See Also

`morlet`, `caps`

Examples

```
library(stats)
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi, prewhiten = FALSE)
Years <- time(ca533.crn)
CAMstd <- ca533.crn[, 1]
out.wave <- morlet(y1 = CAMstd, x1 = Years, p2 = 9, dj = 0.1,
                   siglvl = 0.99)
wavelet.plot(out.wave, useRaster = NA)
## Not run:
# Alternative palette with better separation of colors
# via: rev(RColorBrewer::brewer.pal(10, "Spectral"))
specCols <- c("#5E4FA2", "#3288BD", "#66C2A5", "#ABDDA4", "#E6F598",
              "#FEE08B", "#FDAE61", "#F46D43", "#D53E4F", "#9E0142")
wavelet.plot(out.wave, key.cols=specCols,useRaster = NA)

# fewer colors
levs <- quantile(out.wave$Power, probs = c(0, 0.5, 0.75, 0.9, 0.99))
wavelet.plot(out.wave, wavelet.levels = levs, add.sig = FALSE,
             key.cols = c("#FFFFFF", "#ABDDA4", "#FDAE61", "#D7191C"), useRaster = NA)
```

```
## End(Not run)
```

`wc.to.po` *Convert Wood Completeness to Pith Offset*

Description

This function creates a pith offset data structure based on wood completeness data.

Usage

```
wc.to.po(wc)
```

Arguments

`wc`	A `data.frame` containing wood completeness data, as returned by `read.tridas`.

Details

Computes the sum of the variables *n.missing.heartwood* and *n.unmeasured.inner* in *wc*.

Value

A `data.frame` containing two variables. Variable one (*series*) gives the series ID as either `characters` or `factors`. These match `rownames(`*wc*`)`. Variable two (*pith.offset*) is of `integer` type and gives the years from the beginning of the core to the pith (or center) of the tree. The minimum value is 1.

Author(s)

Mikko Korpela

See Also

`po.to.wc`, `rcs`, `read.tridas`

Examples

```
library(utils)
data(gp.po)
all(wc.to.po(po.to.wc(gp.po)) == gp.po)
```

`write.compact` *Write* DPL *Compact Format Ring Width File*

Description

This function writes a chronology to a DPL compact format file.

Usage

```
write.compact(rwl.df, fname, append = FALSE, prec = 0.01,
              mapping.fname = "", mapping.append = FALSE, ...)
```

Arguments

`rwl.df`	a `data.frame` containing tree-ring ring widths with the series in columns and the years as rows. The series IDs are the column names and the years are the row names. This type of `data.frame` is produced by `read.rwl` and `read.compact`.
`fname`	a `character` vector giving the file name of the rwl file.
`append`	`logical` flag indicating whether to append this chronology to an existing file.
`prec`	`numeric` indicating the precision of the output file. This must be equal to either 0.01 or 0.001 (units are in mm).
`mapping.fname`	a `character` vector of length one giving the file name of an optional output file showing the mapping between input and output series IDs. The mapping is only printed for those IDs that are transformed. An empty name (the default) disables output.
`mapping.append`	`logical` flag indicating whether to append the description of the altered series IDs to an existing file. The default is to create a new file.
`...`	Unknown arguments are accepted but not used.

Details

The output should be readable by the Dendrochronology Program Library (DPL) as a compact format file.

In series IDs, letters of the English alphabet and numbers are allowed. Other characters will be removed. The length of the IDs is limited to about 50 characters, depending on the length of the other items to be placed on the header lines of the output file. Longer IDs will be truncated. Also any duplicate IDs will be automatically edited so that only unique IDs exist. If series IDs are changed, one or more warnings are shown. In that case, the user may wish to print a list of the renamings (see Arguments).

Value

fname

Author(s)

Mikko Korpela, based on write.tucson by Andy Bunn

See Also

write.rwl, write.tucson, write.tridas, read.compact

Examples

```
library(utils)
data(co021)
fname <- write.compact(rwl.df = co021,
                       fname = tempfile(fileext=".rwl"),
                       append = FALSE, prec = 0.001)
print(fname) # tempfile used for output

unlink(fname) # remove the file
```

---

write.crn *Write Tucson Format Chronology File*

Description

This function writes a chronology to a Tucson (decadal) format file.

Usage

```
write.crn(crn, fname, header = NULL, append = FALSE)
```

Arguments

crn	a data.frame containing a tree-ring chronology with two columns of the type produced by chron. The first column contains the mean value chronology, the second column gives the sample depth. Years for the chronology are determined from the row names. The chronology ID is determined from the first column name.
fname	a character vector giving the file name of the crn file.
header	a list giving information for the header of the file. If NULL then no header information will be written.
append	logical flag indicating whether to append this chronology to an existing file.

Details

This writes a standard crn file as defined according to the standards of the ITRDB at https://www1.ncdc.noaa.gov/pub/data/paleo/treering/treeinfo.txt. This is the decadal or Tucson format. It is an ASCII file and machine readable by the standard dendrochronology programs. Header information for the chronology can be written according to the International Tree Ring Data Bank (ITRDB) standard. The header standard is not very reliable however and should be thought of as experimental here. Do not try to write headers using dplR to submit to the ITRDB. When submitting to the ITRDB, you can enter the metadata via their website. If you insist however, the header information is given as a `list` and must be formatted with the following:

Description	*Name*	*Class*	*Max Width*
Site ID	*site.id*	`character`	6
Site Name	*site.name*	`character`	52
Species Code	*spp.code*	`character`	4
State or Country	*state.country*	`character`	13
Species	*spp*	`character`	18
Elevation	*elev*	`character`	5
Latitude	*lat*	`character` or `numeric`	5
Longitude	*long*	`character` or `numeric`	5
First Year	*first.yr*	`character` or `numeric`	4
Last Year	*last.yr*	`character` or `numeric`	4
Lead Investigator	*lead.invs*	`character`	63
Completion Date	*comp.date*	`character`	8

See examples for a correctly formatted header list. If the width of the fields is less than the max width, then the fields will be padded to the right length when written. Note that *lat* and *long* are really *lat*`*100` or *long*`*100` and given as integral values. E.g., 37 degrees 30 minutes would be given as 3750.

This function takes a single chronology with sample depth as input. This means that it will fail if given output from `chron` where *prewhiten* == `TRUE`. However, more than one chronology can be appended to the bottom of an existing file (e.g., standard and residual) with a second call to `write.crn`. However, the ITRDB recommends saving and publishing only one chronology per file. The examples section shows how to circumvent this. The output from this function might be suitable for publication on the ITRDB although the header writing is clunky (see above) and rwl files are much better than crn files in terms of usefulness on the ITRDB.

Value

fname

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

`chron`, `read.crn`

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi)
fname1 <- write.crn(ca533.crn, tempfile(fileext=".crn"))
print(fname1) # tempfile used for output

## Put the standard and residual chronologies in a single file
## with ITRDB header info on top. Not recommended.
ca533.crn <- chron(ca533.rwi, prewhiten = TRUE)
ca533.hdr <- list(site.id = "CAM", site.name = "Campito Mountain",
                  spp.code = "PILO", state.country = "California",
                  spp = "Bristlecone Pine", elev = "3400M", lat = 3730,
                  long = -11813, first.yr = 626, last.yr = 1983,
                  lead.invs = "Donald A. Graybill, V.C. LaMarche, Jr.",
                  comp.date = "Nov1983")
fname2 <- write.crn(ca533.crn[, -2], tempfile(fileext=".crn"),
                    header = ca533.hdr)
write.crn(ca533.crn[, -1], fname2, append = TRUE)
print(fname2) # tempfile used for output

unlink(c(fname1, fname2)) # remove the files
```

`write.rwl` *Write Chronology File*

Description

This function writes a chronology to a file in one of the available formats.

Usage

```
write.rwl(rwl.df, fname, format = c("tucson", "compact", "tridas"), ...)
```

Arguments

`rwl.df`	a `data.frame` containing tree-ring ring widths with the series in columns and the years as rows. The series IDs are the column names and the years are the row names. This type of `data.frame` is produced by `read.rwl`.
`fname`	a `character` vector giving the file name of the rwl file.
`format`	a `character` vector giving the format. This must be `"tucson"`, `"compact"`, or `"tridas"`. Tucson format is the default.
`...`	arguments specific to the function implementing the operation for the chosen format.

Details

This is a simple wrapper to the functions actually implementing the write operation.

Value

fname

Author(s)

Mikko Korpela

See Also

write.crn, write.tucson, write.compact, write.tridas, read.rwl

Examples

```
library(utils)
data(co021)
co021.hdr <- list(site.id = "CO021",
                  site.name = "SCHULMAN OLD TREE NO. 1, MESA VERDE",
                  spp.code = "PSME", state.country = "COLORADO",
                  spp = "DOUGLAS FIR", elev = 2103, lat = 3712,
                  long = -10830, first.yr = 1400, last.yr = 1963,
                  lead.invs = "E. SCHULMAN", comp.date = "")
fname <- write.rwl(rwl.df = co021, fname = tempfile(fileext=".rwl"),
                   format = "tucson", header = co021.hdr,
                   append = FALSE, prec = 0.001)
print(fname) # tempfile used for output

unlink(fname) # remove the file
```

---

write.tridas *Write Tree Ring Data Standard (TRiDaS) file*

Description

This function writes measured or derived (standardized, averaged) series of values to a TRiDaS format file. Some metadata are also supported.

Usage

```
write.tridas(rwl.df = NULL, fname, crn = NULL, prec = NULL, ids = NULL,
    titles = NULL, crn.types = NULL, crn.titles = NULL,
    crn.units = NULL, tridas.measuring.method = NA,
    other.measuring.method = "unknown", sample.type = "core",
    wood.completeness = NULL, taxon = "",
    tridas.variable = "ring width", other.variable = NA,
    project.info = list(type = c("unknown"), description = NULL,
                        title = "", category = "", investigator = "",
                        period = ""),
    lab.info = data.frame(name = "", acronym = NA, identifier = NA,
```

```
                          domain = "", addressLine1 = NA,
                          addressLine2 = NA, cityOrTown = NA,
                          stateProvinceRegion = NA, postalCode = NA,
                          country = NA),
    research.info = data.frame(identifier = NULL, domain = NULL,
                               description = NULL),
    site.info = list(type = "unknown", description = NULL, title = ""),
    random.identifiers = FALSE, identifier.domain = lab.info$name[1],
    ...)
```

Arguments

rwl.df	data.frame containing tree-ring ring widths in millimetres with the series in columns and the years as rows. The series IDs are the column names and the years are the row names. This type of data.frame is produced by read.tucson, read.compact and read.tridas. Defaults to NULL: no measurement series are written.
fname	character vector giving the file name of the rwl file
crn	data.frame or a list of data.frames containing tree-ring chronologies. Accepts data.frames of the type produced by chron. Additionally, allows several chronologies per data.frame. Any column of the data.frame(s) with a name starting with "samp.depth" is interpreted as a sample depth. The rest of the columns are interpreted as chronologies whose titles are determined from the column names (optionally from parameter *crn.titles*). Chronology columns and sample depth columns are paired in order so that the first chronology gets the first sample depth column, second chronology gets the second set of sample depths, etc. If there are less sample depth columns than chronologies, the sample depths are recycled. Defaults to NULL: no chronologies are written.
prec	optional numeric indicating the rounding precision of the output file when writing the data contained in *rwl.df*. Defaults to NULL: no rounding is done and the measurements are written in (non-integer) millimetres. Possible numeric values are 0.001, 0.01, 0.05, 0.1, 1, 10, 100 and 1000, which cause the data to be transformed to micrometres, 1/100th millimetres, 1/20th millimetres, 1/10 millimetres, (millimetres), centimetres, decimetres or metres, respectively, and then rounded to integral values. Data rounded to decimetres are written in centimetres (values always ending in zero). Otherwise, the matching unit is used in the file.
ids	optional data.frame with column one named "tree" giving the numeric ID of the tree, column two named "core" giving the numeric ID of the core, optional column three named "radius" giving the numeric ID of the radius, and optional column four named "measurement" giving the numeric ID of the measurement. If column "measurement" exists, column "radius" must also exist. Defaults to one core, radius and measurement per tree:

```
data.frame(tree=1:n.col, core=rep(1,n.col),
           radius=rep(1,n.col), measurement=rep(1,n.col))
```

where *n.col* is the number of columns in *rwl.df*.

`titles` optional `data.frame` with column one named `"tree"` giving the title of the tree, column two named `"core"` giving the title of the core, column three named `"radius"` giving the title of the radius, and column four named `"measurement"` giving the title of the measurement. By default, *titles* is `NULL`, and the title hierarchy is automatically created out of the column names of *rwl.df*, taking *ids* into account.

`crn.types` `character` vector or a `list` of `character` vectors giving the types of the derived series in *crn*. A single vector is interpreted as one type per `data.frame` in *crn*, recycled if not long enough. A list of vectors is interpreted as one list element per `data.frame`. In this case, the list is recycled to the correct length. After that, the vectors inside the list are recycled to match the number of derived series in each `data.frame` of *crn*. The default is to write empty '`<type>`' elements.

`crn.titles` optional `character` vector or a `list` of `character` vectors giving the titles of the derived series in *crn*. The interpretation is the same as with *crn.types*, except that the default is to derive the titles from the column names of *crn*. Also `NA` means that the column name is used.

`crn.units` optional `character` vector or a `list` of `character` vectors giving the units of the derived series in *crn*. The interpretation is the same as with *crn.types*, except that the default is to mark the series as '`<unitless>`'. Also `NA` means '`<unitless>`'.

`tridas.measuring.method`
`character` vector giving the measuring method used to acquire each series of *rwl.df*. Partial matching is used to replace these with the complete terms in `tridas.vocabulary`. If the vector is shorter than the number of columns in *rwl.df*, it is recycled to the correct length. The default is to use the information in *other.measuring.method* instead. Also, `NA` in any position of the vector means that the measuring method information for that series is looked up in *other.measuring.method*.

`other.measuring.method`
`character` vector giving the measuring method used to acquire each series of *rwl.df*. In contrast to *tridas.measuring.method*, these are free-form strings in English. If the vector is shorter than the number of columns in *rwl.df*, it is recycled to the correct length. The default value is `"unknown"`.

`sample.type` optional `character` vector giving the type of the samples, corresponding to `"core"` in *ids*. The length of the vector, however, must match the number of columns in *rwl.df*, or it is recycled to the correct length. If there are several measurements per sample, some elements of *sample.type* are redundant. The default is to use `"core"` for all series.

`wood.completeness`
optional `data.frame` giving wood completeness information for the measurement series in *rwl.df*. The number of rows must match the number of columns in *rwl.df*. The columns are expected to be a subset of the following (descriptions almost directly quoted from TRiDaS specification):

n.unmeasured.inner Field for recording whether there are any rings at the inner (i.e. towards pith) edge of the sample that have not been measured. Typically used to note when rings are too damaged to measure. Non-negative integral value.

n.unmeasured.outer Field for recording whether there are any rings at the outer (i.e. towards bark) edge of the sample that have not been measured. Typically used to note when rings are too damaged to measure. Non-negative integral value.

pith.presence Whether the pith is present or absent. Each element must be a partial match with the contents of category `"complex presence / absence"` in `tridas.vocabulary`.

heartwood.presence Whether the outer (youngest) heartwood is present and if so whether it is complete.
Category `"complex presence / absence"` in `tridas.vocabulary`.

n.missing.heartwood Estimated number of missing heartwood rings to the pith. Non-negative integral value.

missing.heartwood.foundation Description of the way the estimation of how many heartwood rings are missing was made and what the certainty is. Free-form string.

sapwood.presence Whether the sapwood is present or not.
Category `"complex presence / absence"`.

n.sapwood Number of sapwood rings measured. Non-negative integral value.

last.ring.presence Last ring under the bark is present or absent.
Category `"presence / absence"`.

last.ring.details If the last ring under the bark is present, include information about the completeness of this ring and/or season of felling. Free-form string.

n.missing.sapwood Estimated number of missing sapwood rings to the bark. Non-negative integral value.

missing.sapwood.foundation Description of the way the estimation of how many sapwood rings are missing was made and what the certainty is. Free-form string.

bark.presence Bark is present or absent.
Category `"presence / absence"` in `tridas.vocabulary`.

`taxon` `character` string. The most detailed taxonomic name known for this element (species, genus, family etc). Preferably from the Catalogue of Life controlled vocabulary. The same string is used for all of *rwl.df*. The default value is an empty string, but a proper value should really be given.

`tridas.variable`
`character` string. Measured variable (ring width, earlywood, latewood etc) taken from the TRiDaS controlled vocabulary (`tridas.vocabulary`, category `"variable"`). The same string is used for all of *rwl.df*. Defaults to `"ring width"`.

`other.variable` `character` string. Measured variable as a free-form string. The same string is used for all of *rwl.df*. This is only used if *tridas.variable* is `NA`.

`project.info` `list` containing information about the project. Elements are the following (includes quotes from the TRiDaS specification):

type `character` vector. The type(s) of the project. Defaults to `"unknown"`.

description `character` string. A description of the project. Defaults to `NULL`: no description.

title character string. The title of the project. Defaults to an empty string.

category character string. Category of research this project falls into. Defaults to an empty string.

investigator character string. Principal investigator of this project. Defaults to an empty string.

period character string. When the dendrochronological project took place. Could consist of a start- and end-date. If unknown it should be estimated. Defaults to an empty string.

`lab.info` data.frame. Information about the dendrochronological research laboratories where this work was done. One row per laboratory. Defaults to one laboratory with an empty name and no other information. The columns are expected to be a subset of the following:

name Name of the laboratory

acronym Optional acronym of the laboratory

identifier Optional identifier of the laboratory

domain The domain which the identifier of the laboratory is applicable to. Could be the URL of the organization's server or the name of the organization as long as it is not ambiguous.

addressLine1 First address line

addressLine2 Second address line

cityOrTown City or town

stateProvinceRegion State, province or region

postalCode Postal code. Beware of ignored leading zeros if these are given in numeric or integer type values. It is always safe to use character values.

country Country

`research.info` optional data.frame with information about the systems in which the research project is registered. Columns are the following:

identifier Identifier

domain Domain which the identifier is applicable to

description General description

`site.info` list containing information about the site ('<object>'). Elements are the following, and all are character strings:

type Type of the site. Defaults to "unknown".

description Description. Defaults to no description.

title Title of the site. Defaults to an empty string.

`random.identifiers`

logical flag. If TRUE, unique random identifiers are created with uuid.gen and attached to each '<project>' (one in the file), 'object' (site, one in the file), '<element>' (tree), '<sample>' (core), '<radius>', '<measurementSeries>' (measurement) and '<derivedSeries>' element in the resulting TRiDaS file.

`identifier.domain`

character string. The domain which the random identifiers are applicable to. Could be the URL of the organization's server or the name of the organization as long as it is not ambiguous. Defaults to the name of the first laboratory in *lab.info*.

`...` Unknown arguments are accepted but not used.

Details

The Tree Ring Data Standard (TRiDaS) is described in Jansma et. al (2010).

Value

fname

Note

This is an early version of the function. Bugs are likely to exist, and parameters are subject to change.

Author(s)

Mikko Korpela

References

Jansma, E., Brewer, P. W., and Zandhuis, I. (2010) TRiDaS 1.1: The tree-ring data standard. *Dendrochronologia*, **28**(2), 99–130.

See Also

write.rwl, write.tucson, write.compact, write.crn, read.tridas

Examples

```
library(utils)
## Not run:
## Write raw ring widths
data(co021)
fname1 <- write.tridas(rwl.df = co021,
    fname = tempfile(fileext=".xml"), prec = 0.01,
    site.info = list(title = "Schulman old tree no. 1, Mesa Verde",
                     type = "unknown"),
    taxon = "Pseudotsuga menziesii var. menziesii (Mirb.) Franco",
    project.info = list(investigator = "E. Schulman",
                        title = "", category = "",
                        period = "", type = "unknown"))
print(fname1) # tempfile used for output

## Write mean value chronology of detrended ring widths
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi, prewhiten = TRUE)
fname2 <- write.tridas(crn = ca533.crn,
    fname = tempfile(fileext=".xml"),
    taxon = "Pinus longaeva D.K. Bailey",
    project.info =
        list(investigator = "Donald A. Graybill, V.C. LaMarche, Jr.",
             title = "Campito Mountain", category = "",
```

```
             period = "", type = "unknown"))
print(fname2) # tempfile used for output

unlink(c(fname1, fname2)) # remove the files

## End(Not run)
```

`write.tucson` *Write Tucson Format Chronology File*

Description

This function writes a chronology to a Tucson (decadal) format file.

Usage

```
write.tucson(rwl.df, fname, header = NULL, append = FALSE,
             prec = 0.01, mapping.fname = "", mapping.append = FALSE,
             long.names = FALSE, ...)
```

Arguments

`rwl.df`	a `data.frame` containing tree-ring ring widths with the series in columns and the years as rows. The series IDs are the column names and the years are the row names. This type of `data.frame` is produced by `read.rwl` and `read.compact`.
`fname`	a `character` vector giving the file name of the rwl file.
`header`	a `list` giving information for the header of the file. If `NULL` then no header information will be written.
`append`	`logical` flag indicating whether to append this chronology to an existing file. The default is to create a new file.
`prec`	`numeric` indicating the precision of the output file. This must be equal to either 0.01 or 0.001 (units are in mm).
`mapping.fname`	a `character` vector of length one giving the file name of an optional output file showing the mapping between input and output series IDs. The mapping is only printed for those IDs that are transformed. An empty name (the default) disables output.
`mapping.append`	`logical` flag indicating whether to append the description of the altered series IDs to an existing file. The default is to create a new file.
`long.names`	`logical` flag indicating whether to allow long series IDs (7 or 8 characters) to be written to the output. The default is to only allow 6 characters.
`...`	Unknown arguments are accepted but not used.

Details

This writes a standard rwl file as defined according to the standards of the ITRDB at https://www1.ncdc.noaa.gov/pub/data/paleo/treering/treeinfo.txt. This is the decadal or Tucson format. It is an ASCII file and machine readable by the standard dendrochronology programs. Header information for the rwl can be written according to the International Tree Ring Data Bank (ITRDB) standard. The header standard is not very reliable however and should be thought of as experimental here. Do not try to write headers using dplR to submit to the ITRDB. When submitting to the ITRDB, you can enter the metadata via their website. If you insist however, the header information is given as a `list` and must be formatted with the following:

Description	*Name*	*Class*	*Max Width*
Site ID	*site.id*	`character`	5
Site Name	*site.name*	`character`	52
Species Code	*spp.code*	`character`	4
State or Country	*state.country*	`character`	13
Species	*spp*	`character`	18
Elevation	*elev*	`character`	5
Latitude	*lat*	`character` or `numeric`	5
Longitude	*long*	`character` or `numeric`	5
First Year	*first.yr*	`character` or `numeric`	4
Last Year	*last.yr*	`character` or `numeric`	4
Lead Investigator	*lead.invs*	`character`	63
Completion Date	*comp.date*	`character`	8

See examples for a correctly formatted header list. If the width of the fields is less than the max width, then the fields will be padded to the right length when written. Note that *lat* and *long* are really *lat* `*` `100` or *long* `*` `100` and given as integral values. E.g., 37 degrees 30 minutes would be given as 3750.

Series can be appended to the bottom of an existing file with a second call to `write.tucson`. The output from this file is suitable for publication on the ITRDB.

The function is capable of altering excessively long and/or duplicate series IDs to fit the Tucson specification. Additionally, characters other than numbers or English letters will be removed. If series IDs are changed, one or more warnings are shown. In that case, the user may wish to print a list of the renamings (see Arguments).

Setting *long.names* = `TRUE` allows series IDs to be 8 characters long, or 7 in case there are year numbers using 5 characters. Note that in the latter case the limit of 7 characters applies to all IDs, not just the one corresponding to the series with long year numbers. The default (*long.names* = `FALSE`) is to allow 6 characters. Long IDs may cause incompatibility with other software.

Value

fname

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

write.crn, read.tucson, write.rwl, write.compact, write.tridas

Examples

```
library(utils)
data(co021)
co021.hdr <- list(site.id = "CO021",
                  site.name = "SCHULMAN OLD TREE NO. 1, MESA VERDE",
                  spp.code = "PSME", state.country = "COLORADO",
                  spp = "DOUGLAS FIR", elev = "2103M", lat = 3712,
                  long = -10830, first.yr = 1400, last.yr = 1963,
                  lead.invs = "E. SCHULMAN", comp.date = "")
fname <- write.tucson(rwl.df = co021, fname = tempfile(fileext=".rwl"),
                      header = co021.hdr, append = FALSE, prec = 0.001)
print(fname) # tempfile used for output

unlink(fname) # remove the file
```

---

xdate.floater *Crossdate an undated series*

Description

Pulls an undated series through a dated rwl file in order to try to establish dates

Usage

```
xdate.floater(rwl, series, series.name = "Unk", min.overlap = 50, n = NULL,
              prewhiten = TRUE, biweight = TRUE,
              method = c("spearman", "pearson","kendall"),
              make.plot = TRUE,return.rwl = FALSE, verbose = TRUE)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series.name	a character giving a name for the series.
min.overlap	number
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
prewhiten	logical flag. If TRUE each series is whitened using ar.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.

method	Can be either "pearson", "kendall", or "spearman" which indicates the correlation coefficient to be used. Defaults to "spearman". See cor.test.
make.plot	logical flag indicating whether to make a plot.
return.rwl	logical flag indicating whether to make a plot.
verbose	logical flag indicating whether to print some results to screen.

Details

This takes an undated series (a floater) and drags it along a dated rwl object in order to estabish possible dates for the series. This function is experimental and under development. It might change in future releases.

Value

By default a data.frame is returned with the first and last year of the period compared as well as the correlation, p-value, and number of observations. If return.rwl is TRUE then a list is returned with the rwl object as well as the data.frame with the correlation statistics.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

corr.series.seg, skel.plot, series.rwl.plot, ccf.series.rwl

Examples

```
library(utils)
data(co021)
summary(co021)
foo <- co021[,"645232"]
# 645232  1466 1659
bar <- co021
bar$"645232" <- NULL
out <- xdate.floater(bar, foo, min.overlap = 50, series.name = "Undated")

foo <- co021[,"646118"]
# 646118  1176 1400
bar <- co021
bar$"646118" <- NULL
out <- xdate.floater(bar, foo, min.overlap = 100, series.name = "Undated")
# note that this can fail if a long overlap is needed. This produces the
# wrong dates.
out <- xdate.floater(bar, foo, min.overlap = 200, series.name = "Undated")
```

`xskel.ccf.plot` *Skeleton Plot for Series and Master with Cross Correlation*

Description

...

Usage

```
xskel.ccf.plot(rwl, series, series.yrs = as.numeric(names(series)),
               win.start, win.width = 50, n = NULL,
               prewhiten = TRUE, biweight = TRUE, series.x=FALSE)
```

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`.
`series`	a `numeric` or `character` vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be `numeric`.
`series.yrs`	a `numeric` vector giving the years of *series*. Defaults to `as.numeric(names(`*series*`))`.
`win.start`	year to start window
`win.width`	an even integral value
`n`	`NULL` or an integral value giving the filter length for the `hanning` filter used for removal of low frequency variation.
`prewhiten`	`logical` flag. If `TRUE` each series is whitened using `ar`.
`biweight`	`logical` flag. If `TRUE` then a robust mean is calculated using `tbrm`.
`series.x`	`logical` flag indicating whether to make the `series` the `x` argument to `ccf`. See Details.

Details

This function produces a plot that is a mix of a skeleton plot and a cross-correlation plot. It's used in crossdating.

The top panel shows the normalized values for the master chronology (bottom half) and the series (top half) in green. The values are the detrended and standardized data (e.g., RWI).

Similarly, the black lines are a skeleton plot for the master and series with the marker years annotated for the master on the bottom axis and series on the top. The text at the top of the figure gives the correlation between the series and master (green bars) as well as the percentage of agreement between the years of skeleton bars for the series and master. I.e., if all the black lines occur in the same years the percentage would be 100%.

The bottom panels show cross correlations for the first half (left) and second half of the time series using function `ccf`.

The cross correlations are calculated calling ccf as `ccf(x=master, y=series, lag.max=lag.max, plot=FALSE)` if `series.x` is `FALSE` and as `ccf(x=series, y=master, lag.max=lag.max, plot=FALSE)` if `series.x` is `TRUE`. This argument was introduced in dplR version 1.7.0. Different users have different expectations about how missing or extra rings are notated. If `switch.x = FALSE` the behavior will be like COFECHA where a missing ring in a series produces a negative lag in the plot rather than a positive lag.

The plot is built using the Grid package which allows for great flexibility in building complicated plots. However, these plots look best when they don't cover too wide a range of years (unless the plotting device is wider than is typical). For that reason the user will get a warning if *win.width* is greater than 100 years.

Old-school skeleton plots to print on paper are made with `skel.plot`.

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

ccf

Examples

```
library(utils)
data(co021)
dat <- co021
#corrupt a series
bad.series <- dat$"641143"
names(bad.series) <- rownames(dat)
bad.series <- delete.ring(bad.series,year=1825)
# good match
xskel.ccf.plot(rwl=dat,series=bad.series,win.start=1900,win.width=50)
# overlap missing ring
xskel.ccf.plot(rwl=dat,series=bad.series,win.start=1800,win.width=50)
```

---

xskel.plot *Skeleton Plot for Series and Master*

Description

...

Usage

```
xskel.plot(rwl, series, series.yrs = as.numeric(names(series)),
           win.start, win.end = win.start+100, n = NULL,
           prewhiten = TRUE, biweight = TRUE)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)).
win.start	year to start window
win.end	year to end window
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
prewhiten	logical flag. If TRUE each series is whitened using ar.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.

Details

This function produces a plot that is a mix of a skeleton plot and a cross-correlation plot. It's used in crossdating.

The top panel shows the normalized values for the master chronology (bottom half) and the series (top half) in green. The values are the detrended and standardized data (e.g., RWI).

Similarly, the black lines are a skeleton plot for the master and series with the marker years annotated for the master on the bottom axis and series on the top. The text at the top of the figure gives the correlation between the series and master (green bars) as well as the percentage of agreement between the years of skeleton bars for the series and master. I.e., if all the black lines occur in the same years the percentage would be 100%.

The bottom panels show cross correlations for the first half (left) and second half of the time series using function ccf as ccf(x=series,y=master,lag.max=5).

The plot is built using the Grid package which allows for great flexibility in building complicated plots. However, these plots look best when they don't cover too wide a range of years (unless the plotting device is wider than is typical). For that reason the user will get a warning if *win.width* is greater than 100 years.

Old-school skeleton plots to print on paper are made with skel.plot.

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

ccf

Examples

```
library(utils)
data(co021)
dat <- co021
#corrupt a series
bad.series <- dat$"641143"
names(bad.series) <- rownames(dat)
bad.series <- delete.ring(bad.series,year=1825)
# good match
xskel.plot(rwl=dat,series=bad.series,win.start=1850)
# overlap missing ring
xskel.plot(rwl=dat,series=bad.series,win.start=1800)
```

---

zof.anc *Ancillary Data Corresponding to* zof.rwl

Description

This data set gives the pith offsets, distance to pith, and diameter that match the ring widths for zof.rwl – a data set of European Beech (*Fagus sylvatica*) increment cores collected at the Zofingen in Switzerland.

Usage

```
data(zof.anc)
```

Format

A data.frame containing four columns. Column one gives the series ID. Column two (PO) gives the number of rings estimated to be missing to the pith. Column three (d2pith) gives the estimated distance to the pith (mm). Column four (diam) gives the diameter at breast height (DBH) in cm.

Source

Contributed by Stefan Klesse

References

Klesse, S., Babst, F., Lienert, S., Spahni, R., Joos, F., Bouriaud, O., Carrer, M., Filippo, A.D., Poulter, B., Trotsiuk, V., Wilson, R., Frank, D.C. (2018) A combined tree ring and vegetation model assessment of European forest growth sensitivity to interannual climate variability. *Global Biogeochemical Cycles* 32, 1226 – 1240.

`zof.rwl` *European Beech Ring Widths from Zofingen, Switzerland*

Description

This data set includes ring-width measurements for European Beech (*Fagus sylvatica*) increment cores collected at the Zofingen in Switzerland. There are 61 series.

Usage

```
data(zof.rwl)
```

Format

A `data.frame` containing 61 ring-width series in columns and 156 years in rows.

Source

Contributed by Stefan Klesse

References

Klesse, S., Babst, F., Lienert, S., Spahni, R., Joos, F., Bouriaud, O., Carrer, M., Filippo, A.D., Poulter, B., Trotsiuk, V., Wilson, R., Frank, D.C. (2018) A combined tree ring and vegetation model assessment of European forest growth sensitivity to interannual climate variability. *Global Biogeochemical Cycles* 32, 1226 – 1240.

Index