

Package ‘dplr’

July 2, 2014

Encoding UTF-8

Type Package

Title Dendrochronology Program Library in R

Version 1.6.0

Date 2014-04-24

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

Copyright Authors and Aalto University (for work of M. Korpela)

Maintainer Andy Bunn <andy.bunn@wwu.edu>

Depends R (>= 2.15.0)

Imports gmp (>= 0.5-2), graphics, grDevices, grid, stats, utils,digest (>= 0.2.3), lattice (>= 0.13-6), stringr (>= 0.4), XML (>= 2.1-0)

Suggests foreach, forecast, iterators, RUnit (>= 0.4.25), waveslim

Description This package contains functions for performing tree-ring analyses, IO, and graphics.

LazyData no

License GPL (>= 2)

URL <http://huxley.wwu.edu/trl/htrl-dplr>,
<http://R-Forge.R-project.org/projects/dplr/>

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-04-25 22:44:57

R topics documented:

dpIR-package	3
anos1	4
bai.in	5
bai.out	6
ca533	8
cana157	9
ccf.series.rwl	10
chron	12
cms	13
co021	15
combine.rwl	15
common.interval	16
corr.rwl.seg	17
corr.series.seg	19
crn.plot	21
detrend	23
detrend.series	25
ffcsaps	29
fill.internal.NA	30
gini.coef	32
glk	33
gp.d2pith	34
gp.dbh	34
gp.po	35
gp.rwl	36
hanning	36
i.detrend	37
i.detrend.series	38
insert.ring	39
interseries.cor	40
morlet	42
plot.rwl	43
po.to.wc	44
pointer	45
powt	46
print.redfit	47
rsc	49
read.compact	51
read.crn	52
read.fh	53
read.ids	54
read.rwl	57
read.tridas	58
read.tucson	65
redfit	66
rwi.stats.running	71

rw1.stats	76
sea	77
seg.plot	78
sens1	79
sens2	80
series.rwl.plot	81
skel.plot	83
spag.plot	85
strip.rwl	86
tbrm	87
tridas.vocabulary	88
uuid.gen	90
wavelet.plot	92
wc.to.po	94
write.compact	95
write.crn	96
write.rwl	98
write.tridas	99
write.tucson	105
xskel.ccf.plot	107
xskel.plot	108

Index**111**

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

Main Functions

[read.rwl](#) reads rw1 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 [detrrend.series](#).

References

Cook, E. R. and Kairiukstis, L. A. (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.

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. PhD-Thesis, Technische Universität München.

`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

<code>rwl</code>	a <code>data.frame</code> with series as columns and years as rows such as that produced by read.rwl
<code>d2pith</code>	an optional vector 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 <code>colnames(rwl)</code> . Variable two (<code>d2pith</code> in the example below) gives the distance from the innermost measured ring to the pith of the tree in mm. If <code>d2pith</code> is <code>NULL</code> then the distance to pith is assumed to be zero for each series (column) in <code>rwl</code> .

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: <http://dendrolab.org/>

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)
## 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)
yr <- as.numeric(rownames(foo.crn))
plot(yr, foo.crn[, 1], type = "n",
     xlab = "Year", ylab = expression(mm^2))
lines(yr, foo.crn[, 1], col = "grey", lty = "dashed")
lines(yr, ffcaps(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

<code>rwl</code>	a <code>data.frame</code> with series as columns and years as rows such as that produced by read.rwl
<code>diam</code>	an optional <code>data.frame</code> containing two variables. If present, variable one (<i>series</i> in the example below) gives the series ID as either characters or factors. These must exactly match <code>colnames(rwl)</code> . Variable two (<i>diam</i> 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 <i>diam</i> is <code>NULL</code> then the diameter is taken as twice the sum of the widths for each series (column) in <i>rwl</i> .

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: <http://dendrolab.org/>

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

```
## Not run:
library(graphics)
## 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)
yr <- as.numeric(rownames(bar.crn))
plot(yr, bar.crn[, 1], type = "n",
     xlab = "Year", ylab = expression(mm^2))
lines(yr, bar.crn[, 1], col = "grey", lty = "dashed")
lines(yr, ffcaps(bar.crn[, 1], nyrs = 32), col = "red", lwd = 2)
```

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 27-August-2007 at <ftp://ftp.ncdc.noaa.gov/pub/data/paleo/treering/measurements/northamerica/usa/ca533.rwl>

References

Graybill, D. (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 27-August-2007 at <ftp://ftp.ncdc.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.

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, ...)
```

Arguments

<code>rwl</code>	a data.frame with series as columns and years as rows such as that produced by read.rwl .
<code>series</code>	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of <code>rwl</code> is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
<code>series.yrs</code>	a numeric vector giving the years of <code>series</code> . Defaults to <code>as.numeric(names(series))</code> . Ignored if <code>series</code> is an index to a column of <code>rwl</code> .
<code>seg.length</code>	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
<code>bin.floor</code>	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.
<code>n</code>	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
<code>prewhiten</code>	logical flag. If TRUE each series is whitened using ar .
<code>biweight</code>	logical flag. If TRUE then a robust mean is calculated using tbrm .
<code>pcrit</code>	a number between 0 and 1 giving the critical value for the correlation test.
<code>lag.max</code>	an integral value giving the maximum lag at which to calculate the ccf .
<code>make.plot</code>	logical flag indicating whether to make a plot.
<code>floor.plus1</code>	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
<code>...</code>	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=series, y=master, lag.max=lag.max, plot=FALSE)`. Note that prior to `dplR` version 1.60, the master was set as `x` and the series as `y`. This was changed to be more in line with user expectations so that a missing ring in a series produces a positive lag in the plot rather than a negative lag. This structure of this call does put the plots at odds with Figure 3 in Bunn (2010) which is unfortunate.

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 AG (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

```
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 [detrnd](#).

Usage

```
chron(x, prefix = "xxx", biweight = TRUE, prewhiten = FALSE)
```

Arguments

<code>x</code>	a <code>data.frame</code> of ring widths with <code>rownames(x)</code> containing years and <code>colnames(x)</code> containing each series ID such as produced by read.rwl
<code>prefix</code>	a character string with less than 4 characters. Defaults to "xxx"
<code>biweight</code>	logical flag. If <code>TRUE</code> then a robust mean is calculated using tbrm .
<code>prewhiten</code>	logical flag. If <code>TRUE</code> each series is whitened using ar prior to averaging.

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).

Value

A `data.frame` with the standard chronology, residual chronology (if prewhitening was performed), and the sample depth.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Cook, E. R. and Kairiukstis, L. A. (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

```
data(ca533)
ca533.rwl <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwl, prefix = "CAM")
plot(ca533.crn,xlab="Year",ylab="RWI")
## With residual chron
ca533.crn <- chron(ca533.rwl, prefix = "CAM", prewhiten = TRUE)
```

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

<code>rwl</code>	a <code>data.frame</code> with series as columns and years as rows such as that produced by read.rwl
<code>po</code>	a <code>data.frame</code> containing two variables. Variable one (<i>series</i> in the example below) gives the series ID as either characters or factors. These must exactly match <code>colnames(rwl)</code> . Variable two (<i>pith.offset</i> 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.
<code>c.hat.t</code>	a logical indicating whether to export the C-curves for each tree by biological age.
<code>c.hat.i</code>	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 *rw1* 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: <http://dendrolab.org/>

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](#), [rccs](#)

Examples

```
library(graphics)
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 12-September-2011 at <ftp://ftp.ncdc.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

A `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

```
data(ca533)
data(co021)
combine.rwl(list(ca533, co021))
## or alternatively for data.frames to combine
combine.rwl(ca533, co021)
```

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

<code>rwl</code>	a <code>data.frame</code> of ring widths with <code>rownames(x)</code> containing years and <code>colnames(x)</code> containing each series ID such as produced by read.rwl
<code>type</code>	a character string of "series", "years", or "both". Argument matching is performed.
<code>make.plot</code>	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

```
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

<code>rwl</code>	a <code>data.frame</code> with series as columns and years as rows such as that produced by read.rwl .
<code>seg.length</code>	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
<code>bin.floor</code>	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.
<code>n</code>	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
<code>prewhiten</code>	logical flag. If TRUE each series is whitened using ar .
<code>pcrit</code>	a number between 0 and 1 giving the critical value for the correlation test.
<code>biweight</code>	logical flag. If TRUE then a robust mean is calculated using tbrm .
<code>method</code>	Can either "pearson", "kendall", or "spearman" which indicates the correlation coefficient is to be used. Defaults to "spearman." See cor.test .
<code>make.plot</code>	logical flag indicating whether to make a plot.
<code>label.cex</code>	numeric scalar for the series labels on the plot. Passed to <code>axis.cex</code> in axis .
<code>floor.plus1</code>	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
<code>master</code>	NULL, a numeric vector or a matrix-like object of numeric values, including a <code>data.frame</code> . If NULL, a number of master chronologies, one for each series in <code>rwl</code> , is built from <code>rwl</code> using the leave-one-out principle. If a vector, the function uses this as the master chronology. If a matrix or <code>data.frame</code> , this object is used for building the master chronology (no leave-one-out).
<code>master.yrs</code>	a numeric vector giving the years of <i>series</i> . Defaults to names or rownames of <code>master</code> coerced to numeric type.
<code>...</code>	other arguments passed to <code>plot</code> .

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 $\text{ceiling}(\text{min.yr}/\text{bin.floor}) * \text{bin.floor}$ 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 $\text{floor}(n/2)$. 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*, vector *avg.seg.rho*. 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.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

[corr.series.seg](#), [skel.plot](#), [series.rwl.plot](#), [ccf.series.rwl](#)

Examples

```
data(co021)
corr.rwl.seg(co021, seg.length = 100, label.cex = 1.25)
```

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

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

Details

This function calculates the correlation 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 each series 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

```
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)
```

crn.plot

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](#).

Usage

```
crn.plot(crn, add.spline = FALSE, nyrs = NULL, f = 0.5,
        crn.line.col='grey50', spline.line.col='red',
        samp.depth.col='grey90',
        samp.depth.border.col='grey80',
        crn.lwd=1, spline.lwd=1.5,
        abline.pos=1, abline.col='black',
        abline.lty=1, abline.lwd=1, ...)
```

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

Arguments

<code>x</code> , <code>crn</code>	a data.frame e.g., as produced by chron . The data.frame should have the years in <code>rownames(crn)</code> , the chronologies in the columns. Optionally, the last column can contain the sample depth named "samp.depth". If the data are produced via chron the object will be of class "crn" and the generic plot method can be used
<code>add.spline</code>	a logical flag. Will add a line with a smoothing spline using ffcsaps
<code>nyrs</code>	a number giving the rigidity of the smoothing spline. Defaults to 0.33 times the length of the first chronology if <code>nyrs</code> is NULL
<code>f</code>	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5
<code>crn.line.col</code>	color for the x line
<code>spline.line.col</code>	color for the spline (if added)
<code>samp.depth.col</code>	color for the sample depth polygon (if present)
<code>samp.depth.border.col</code>	border color for the sample depth polygon (if present)
<code>crn.lwd</code>	line width for the x line
<code>spline.lwd</code>	line width for the spline (if added)
<code>abline.pos</code>	position for a reference abline on the y-axis. No line added if NULL
<code>abline.col</code>	color for the reference abline (if added)
<code>abline.lty</code>	line type the reference abline (if added)
<code>abline.lwd</code>	line width for the reference abline (if added)
<code>...</code>	Additional arguments to pass to plot

Details

This makes a 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

```

data(cana157)
crn.plot(cana157,xlab='Year',ylab='RWI')
plot(cana157,xlab='Year',ylab='RWI')
# with added spline
crn.plot(cana157,add.spline=TRUE, nyrs=32, xlab='Year',ylab='RWI')
## Without sample depth
cana157.mod <- cana157
cana157.mod$samp.depth <- NULL
crn.plot(cana157.mod, add.spline = TRUE, xlab='Year',ylab='RWI')
## With multiple chronologies
data(gp.rwl)
data(gp.po)
gp.rwi <- cms(rwl = gp.rwl, po = gp.po)
gp.crn <- chron(gp.rwi,prefix="GP",prewhiten=TRUE)
crn.plot(gp.crn, add.spline = TRUE)
## Not run:
# not pretty - but illustrates the coloring options
my.cols <- c("#3182BD", "#9ECAE1", "#DEEBF7", "#31A354", "#A1D99B", "#E5F5E0")
crn.plot(cana157,add.spline=TRUE,nyrs=32,
        crn.line.col=my.cols[5],
        spline.line.col=my.cols[4],
        samp.depth.col=my.cols[3],
        samp.depth.border.col=my.cols[2],
        abline.col=my.cols[1],
        crn.lwd=1.5,spline.lwd=3,
        abline.lwd=1)
# a raw ring-width chronology
data(ca533)
ca533.raw.crn <- chron(ca533, prefix = "CAM")
crn.plot(ca533.raw.crn,abline.pos=NULL,ylab='mm')

## End(Not run)

```

detrend

Detrend Multiple Ring-Width Series Simultaneously

Description

This is a wrapper for [detrend.series](#) to detrend many ring-width series at once.

Usage

```
detrrend(rwl, y.name = names(rwl), make.plot = FALSE,
         method = c("Spline", "ModNegExp", "Mean", "Ar"), nyrs = NULL,
         f = 0.5, pos.slope = FALSE,
         constrain.modnegexp = c("never", "when.fail", "always"),
         verbose = FALSE, return.info = FALSE)
```

Arguments

<code>rwl</code>	a <code>data.frame</code> with series as columns and years as rows such as that produced by read.rwl
<code>y.name</code>	a character vector of length(<code>ncol(rwl)</code>) that gives the ID of each series. Defaults to the column names of <code>rwl</code> .
<code>make.plot</code>	a logical flag. Makes plots of the raw data and detrended data if TRUE. See details below.
<code>method</code>	a character vector to determine the detrending methods. See details below. Possible values are all subsets of <code>c("Spline", "ModNegExp", "Mean", "Ar")</code> . Defaults to using all the available methods.
<code>nyrs</code>	a number giving the rigidity of the smoothing spline, defaults to 0.67 of series length if <code>nyrs</code> is NULL.
<code>f</code>	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
<code>pos.slope</code>	a logical flag. Will allow for a positive slope to be used in method "ModNegExp". If FALSE the line will be horizontal.
<code>constrain.modnegexp</code>	a character string which controls the constraints of the "ModNegExp" model. See detrrend.series for further details.
<code>verbose</code>	logical. Write out details?
<code>return.info</code>	a logical flag. If TRUE, details about models and data will be added to the return value. See 'Value'.

Details

See [detrrend.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 three parts:

<code>series</code>	the main result described above (<code>data.frame</code> or list of <code>data.frames</code>)
---------------------	---

model.info	Information about the models corresponding to each output series. A list with one element for each column of <i>rwl</i> . See detrend.series ('Value', <i>model.info</i>) for a description of the contents.
data.info	Information about the input series. A list with one element for each column of <i>rwl</i> . See detrend.series ('Value', <i>data.info</i>).

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

```
data(ca533)
## Detrend using modified exponential decay. Returns a data.frame
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")

## 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"),
              nyrs = NULL, f = 0.5, pos.slope = FALSE,
              constrain.modnegexp = c("never", "when.fail", "always"),
              verbose=FALSE, return.info=FALSE)
```

Arguments

<code>y</code>	a numeric vector. Usually a tree-ring series.
<code>y.name</code>	an optional character vector to name the series for plotting purposes.
<code>make.plot</code>	a logical flag. Makes plots of the raw data and detrended data if TRUE.
<code>method</code>	a character vector to determine the detrending methods. See details below. Possible values are all subsets of <code>c("Spline", "ModNegExp", "Mean", "Ar")</code> . Defaults to using all the available methods.
<code>nyrs</code>	a number giving the rigidity of the smoothing spline, defaults to 0.67 of series length if <code>nyrs</code> is NULL.
<code>f</code>	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
<code>pos.slope</code>	a logical flag. Will allow for a positive slope to be used in method "ModNegExp". If FALSE the line will be horizontal.
<code>constrain.modnegexp</code>	a character string which controls the constraints of the "ModNegExp" model. The value is an answer to the question: When should the parameters of the modified negative exponential 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'.
<code>verbose</code>	a logical flag. Write out details?
<code>return.info</code>	a logical flag. If TRUE, details about models and data will be added to the return value. See 'Value'.

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 standardization is done by dividing each series by the growth trend to produce units in the dimensionless ring-width index (RWI). There are currently three methods available for detrending although more are certainly possible. The methods implemented are a smoothing spline via `ffcsaps` (`method = "Spline"`), a modified negative exponential curve (`method = "ModNegExp"`), or a simple horizontal line (`method = "Mean"`).

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

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.modnegexp* 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.modnegexp* = "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 "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 "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.

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.

See the references below for further details on detrending.

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 three parts:

<code>series</code>	the main result described above (<code>data.frame</code> or vector)
<code>model.info</code>	Information about the models corresponding to each output series. Whereas <i>series</i> may return a vector, <i>model.info</i> 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 <i>constrain.modnegexp</i> is set to "when.fail".

nyrs The value of *nyrs* used for `ffcsaps`. Only for method "Spline".

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.

Author(s)

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

References

Cook, E. R. and Kairiukstis, L. A. (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)
## Using a plausible representation of a tree-ring series
gt <- 0.5 * exp (-0.05 * 1:200) + 0.2
noise <- c(arima.sim(model = list(ar = 0.7), n = 200, sd = 0.5))+2
series <- gt * noise
series.rwi <- detrend.series(y = series, y.name = "Foo", verbose=TRUE)
## Use series CAM011 from the Campito data set
data(ca533)
series <- ca533[, "CAM011"]
names(series) <- rownames(ca533)
# defaults to all four 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", "Mean"))
# 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
series.rwi <- detrend.series(y = series, y.name = "CAM011",
                           method="Spline")
# 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")
```

ffcsaps*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

```
ffcsaps(y, x = seq_along(y), nyrs = length(y)/2, f = 0.5)
```

Arguments

y	a numeric vector, typically a tree-ring series.
x	a numeric vector with the same length as y , typically giving the years of y .
$nyrs$	a number greater than 1, affecting the rigidity of the spline. When f is kept constant, a larger $nyrs$ produces a more rigid spline. Defaults to $\text{length}(y)/2$.
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, $f = 1$ results in the natural spline, i.e. the function outputs y . The default value is 0.5.

Details

This applies a smoothing spline similar to the spline applied in most dendrochronological software. See references for more information.

Value

A filtered vector.

Note

DendroLab website: <http://dendrolab.org/>

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

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

See Also

[hanning](#), [detrend](#)

Examples

```
## Not run:
library(graphics)
## Use series CAM011 from the Campito data set
data(ca533)
series <- ca533[, "CAM011"]
series <- series[!is.na(series)]
plot(series, type = "l", ylab = "Ring Width (mm)", col = "grey")
lines(ffcsaps(series, nyrs = 32), col = "red", lwd = 2)
lines(ffcsaps(series, nyrs = 64), col = "green", lwd = 2)
lines(ffcsaps(series, nyrs = 128), col = "blue", lwd = 2)

## End(Not run)
## 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(ffcsaps(series, nyrs = 32), col = "red", lwd = 2)
lines(ffcsaps(series, nyrs = 64), col = "green", lwd = 2)
## nyrs defaults to 0.5*length(series) == 347
lines(ffcsaps(series), col = "blue", lwd = 2)
legend("topright",
      c("Series", "nyrs=32", "nyrs=64",
        paste("Default nyrs (", length(series) / 2, ")", sep="")),
      fill=c("grey", "red", "green", "blue"))
```

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

<code>x</code>	a data.frame of ring widths with <code>row.names(x)</code> containing years and <code>names(x)</code> containing each series ID such as produced by <code>read.rwl</code>
<code>fill</code>	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. `ffcsaps`). 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)
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
```

```
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 <- as.numeric(row.names(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-Verlag. ISBN: 0-387-98778-9.

See Also[rwl.stats](#)**Examples**

```
data(ca533)
ca533.rwl <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwl, prefix = "CAM")
gini.coef(ca533.crn)
```

glk*Calculate Gleichläufigkeit*

Description

This function calculates the matrix of Gleichläufigkeit for a given set of tree-ring records.

Usage

```
glk(x)
```

Arguments

`x` a `data.frame` of tree-ring data with records in columns, and years as rows.

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, and thus returns a matrix whose upper triangle reports the Gleichläufigkeit of each possible combination of records. The global Gleichläufigkeit can be calculated as `mean(glk(x), na.rm = TRUE)`.

Value

A matrix with the Gleichläufigkeit for all possible combinations of records. If two curves have less than 3 years of overlap, Gleichläufigkeit cannot be computed, and NA is returned.

Author(s)

Christian Zang. Patched and improved by Mikko Korpela.

References

Eckstein, D., Bauch J. (1969) Beitrag zur Rationalisierung eines dendrochronologischen Verfahrens und zur Analyse seiner Aussagesicherheit. *Forstwissenschaftliches Centralblatt*, 88(1):230–250.
Schweingruber, F. H. (1988) *Tree rings: basics and applications of dendrochronology*. Kluwer Academic Publishers, Dordrecht, Netherlands, 276 p.

Examples

```
data(ca533)
ca533.g1k <- g1k(ca533)
mean(ca533.g1k, na.rm = TRUE)
```

gp.d2pith	<i>Ponderosa Pine Distance to Pith Corresponding to gp.rwl</i>
-----------	--

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. <http://dendrolab.org/>

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	<i>Ponderosa Pine Stem Diameters and Bark Thickness (gp.rwl)</i>
--------	--

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. <http://dendrolab.org/>

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. <http://dendrolab.org/>

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. <http://dendrolab.org/>

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**

<code>x</code>	a vector
<code>n</code>	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 (2nd Edition)*. Prentice-Hall. ISBN-13: 978-0-13-754920-7.

See Also

[filter](#)

Examples

```
library(graphics)
data(ca533)
yrs <- as.numeric(rownames(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

rw1	a data.frame with series as columns and years as rows such as that produced by read.rw1 or ca533
y.name	a character vector of length ncol(rw1) that gives the ID of each series. Defaults to the column names of rw1.
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 rw1. 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

<code>y</code>	a numeric vector. Usually a tree-ring series.
<code>y.name</code>	an optional character vector to name the series for plotting purposes.
<code>nyrs</code>	a number giving the rigidity of the smoothing spline, defaults to 0.67 of series length if <code>nyrs</code> is NULL.
<code>f</code>	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
<code>pos.slope</code>	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)
delete.ring(rw.vec,rw.vec.yrs=as.numeric(names(rw.vec)),
            year,fix.last=TRUE)
```

Arguments

<code>rw.vec</code>	a vector of data
<code>rw.vec.yrs</code>	the years for <code>rw.vec</code> as names
<code>year</code>	the year to add or delete
<code>ring.value</code>	the value to add
<code>fix.last</code>	logical. If TRUE the last year of the series is fixed and the first year changes.

Details

Simple editing of ring widths.

Value

A named vector.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

[dpLR](#)

Examples

```
data(gp.rwl)
series <- gp.rwl$"50A"
names(series) <- rownames(gp.rwl)
series <- series[!is.na(series)]
series
insert.ring(rw.vec=series,year=1950,ring.value=0)
insert.ring(rw.vec=series,year=1950,ring.value=0,fix.last=FALSE)
delete.ring(rw.vec=series,year=1900)
delete.ring(rw.vec=series,year=1900,fix.last=FALSE)
```

<code>interseries.cor</code>	<i>Calculate an individual series correlation against a master chronology in an <code>rwl</code> object</i>
------------------------------	---

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

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

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

```
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])
```

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 available at <http://atoc.colorado.edu/research/wavelets/software.html>

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

```
data(ca533)
ca533.rwl <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwl, prefix = "CAM", prewhiten = FALSE)
Years <- as.numeric(rownames(ca533.crn))
CAMstd <- ca533.crn[, 1]
out.wave <- morlet(y1 = CAMstd, x1 = Years, dj = 0.1, siglvl = 0.99)
```

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**

```
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:
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

<code>rwl</code>	a <code>data.frame</code> with ring-width series as columns and years as rows such as that produced by <code>read.rwl</code> .
<code>rgv.thresh</code>	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.
<code>nseries.thresh</code>	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.
<code>round.decimals</code>	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 Géréma, 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.
data(gp.rwl)
pointer(rwl=gp.rwl, rgv.thresh=10, nseries.thresh=75, round.decimals=2)
```

powt

Power Transformation of Tree-Ring Data

Description

Perform power transformation simultaneously for a `data.frame` of raw tree-ring width series.

Usage

```
powt(rwl)
```

Arguments

rw1 a data.frame of raw tree-ring widths series, such as that produced by [read.rwl](#) or [read.fh](#)

Details

This procedure 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 $\log S = k + b \log M$ is then used for the power transform $\star R_t = R_t^{1-b}$.

Value

A data.frame containing the power transformed ring width series.

Author(s)

Christian Zang. Patched and improved by Mikko Korpela.

References

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

See Also

[rcs](#)

Examples

```
data(gp.rwl)
gp.pt <- powt(gp.rwl)
```

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

<code>x</code>	An object of class "redfit".
<code>digits</code>	Specifies the desired number of significant digits in the output. The argument is passed to <code>format</code> and <code>print.data.frame</code> . A positive integral value or NULL. If NULL, the value in <code>options("digits")</code> is used.
<code>csv.out</code>	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'.
<code>do.table</code>	A logical flag. If TRUE, the large information table is also printed when <code>csv.out</code> is FALSE, although not in a comma-separated format but with <code>print.data.frame</code> .
<code>prefix</code>	A prefix to be used on every output line except the large informaton table. REDFIT (see References) uses "# ".
<code>row.names</code>	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.
<code>file</code>	A writable connection or a character string naming a file. Used for setting the output destination when <code>csv.out</code> is TRUE. The default is to write the comma-separated table to the console.
<code>...</code>	Arguments to <code>write.csv</code> . Used when <code>csv.out</code> 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

```
data(ca533)
t <- as.numeric(row.names(ca533))
x <- ca533[[1]]
idx <- which(!is.na(x))
redf <- redfit(x[idx], t[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

```

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, ...)

```

Arguments

<code>rwl</code>	a data.frame with series as columns and years as rows such as that produced by read.rwl
<code>po</code>	a data.frame containing two variables. Variable one (<i>series</i> in the example below) gives the series ID as either characters or factors. These must exactly match <code>colnames(rwl)</code> . Variable two (<i>pith.offset</i> 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.
<code>nyrs</code>	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 <code>nyrs</code> is NULL.
<code>f</code>	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
<code>biweight</code>	logical flag. If TRUE then a robust mean is calculated using tbrm .
<code>r ratios</code>	logical flag. If TRUE (the default) then indices are calculated by division, if FALSE indices are calculated by subtraction.
<code>make.plot</code>	logical flag. Makes plots of the raw data and regional curve if TRUE.
<code>rc.out</code>	logical flag. Returns the regional curve along with the ring-width indices if TRUE.
<code>...</code>	other arguments passed to plot .

Details

This method detrends and standardizes tree-ring series by calculating an age-related growth curve specific to the *rw1*. 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 [ffcsaps](#).

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 *rw1*. 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: <http://dendrolab.org/>

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. (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](#), [ffcsaps](#)

Examples

```
data(gp.rw1)
data(gp.po)
gp.rwi <- rcs(rw1 = gp.rw1, 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

A 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](#)

read.crn	<i>Read Tucson Format Chronology File</i>
----------	---

Description

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

Usage

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

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 .

Details

This reads in a standard crn file as defined according to the standards of the ITRDB at <http://www.ncdc.noaa.gov/paleo/treeinfo.html>. 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)
```

Arguments

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

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

A `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 [rccs](#).

Author(s)

Christian Zang. New features and patches by Mikko Korpela.

References

Rinn, F. (2003) *TSAP-Win User Reference Manual*. Rinntech, Heidelberg <http://www.rinntech.com/>

See Also

[read.rwl](#)

read.ids	<i>Read Site-Tree-Core IDs</i>
----------	--------------------------------

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

<code>rwl</code>	a data.frame with series as columns and years as rows such as that produced by read.rwl or ca533
<code>stc</code>	a vector of three integral values or character string "auto". The numbers indicate the number of characters to split the site code (<code>stc[1]</code>), the tree IDs (<code>stc[2]</code>), and the core IDs (<code>stc[3]</code>). Defaults to <code>c(3, 2, 3)</code> . If "auto", tries to automatically determine the split locations. See Details for further information.
<code>use.cor</code>	a logical flag. If TRUE and <code>stc</code> 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 <code>read.ids</code> and <code>autoread.ids</code> .
<code>ignore.site.case</code>	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.
<code>ignore.case</code>	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 <code>read.ids</code> is FALSE, i.e. the difference matters. The default in <code>read.ids</code> 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 <code>stc</code> 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.
<code>fix.typos</code>	a logical flag. If TRUE, the function will try to detect and fix typing errors.
<code>typo.ratio</code>	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).
 - If 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 $\text{typo.ratio} / (\text{typo.ratio} + 1) * n.tot$, 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 $\text{typo.ratio} / (\text{typo.ratio} + 1)$. 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

```
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"),
         ...)
```

Arguments

fname	a character vector giving the file name of the rwl file.
format	a character vector giving the format. This must be "auto" (automatic detection), "tucson", "compact", "tridas" or "heidelberg". Automatic format detection 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 read operation.

Value

If a "tucson" or "compact" file is read (even through "auto"), returns a `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](#), [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

<code>fname</code>	character vector giving the file name of the TRiDaS file.
<code>ids.from.titles</code>	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.
<code>ids.from.identifiers</code>	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 <code>ids.from.titles == TRUE</code> . Defaults to TRUE.
<code>combine.series</code>	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 <code>data.frame</code> . 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 <code>ids.from.titles ids.from.identifiers</code> .
<code>trim.whitespace</code>	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.
<code>warn.units</code>	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 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:

<code>measurements</code>	A <code>data.frame</code> or a list of <code>data.frames</code> 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 <i>measurements</i> . If <code>!combine.series</code> && <code>(ids.from.titles ids.from</code> 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 <i>measurements</i> .
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 <i>measurements</i> .
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 <i>measurements</i> . Length equals the number of data.frames.
project.title	A character vector giving the title of the project of each data.frame in <i>measurements</i> . 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 <i>measurements</i> . 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 <i>measurements</i> . May have several columns to reflect the possibly nested <object> elements.
taxon	A data.frame showing the taxonomic name for each data.frame in <i>measurements</i> . 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 <i>measurements</i> . 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 <code>undated\$data</code> ids A data.frame with columns named "tree", "core", "radius", and "measurement", together giving a numeric ID for each measurement series in <code>undated\$data</code> .

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:

	<p>text The text of the type element</p> <p>lang The language of the text</p> <p>normal The normalized name of the type</p> <p>normalId The ID value of the type in the standard dictionary</p> <p>normalStd The name of the standard</p> <p>project.id The ID of the project</p> <p>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.</p> <p>tree.id The ID of the tree</p> <p>core.id The ID of the core</p> <p>derived.id The ID of the derived series</p> <p>project.title The title of the project</p> <p>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.</p> <p>tree.title The title of the tree</p> <p>core.title The title of the core</p> <p>derived.title The title of the derived series</p>
comments	<p>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:</p> <p>text The text of the comments element</p> <p>project.id The ID of the project</p> <p>site.id One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the ID of the site.</p> <p>tree.id The ID of the tree</p> <p>core.id The ID of the core</p> <p>radius.id The ID of the radius</p> <p>measurement.id The ID of the measurement series</p> <p>derived.id The ID of the derived series</p> <p>project.title The title of the project</p> <p>site.title One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the title of the site.</p> <p>tree.title The title of the tree</p> <p>core.title The title of the core</p> <p>radius.title The title of the radius</p> <p>measurement.title The title of the measurement series</p> <p>derived.title The title of the derived series</p>
identifier	<p>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:</p> <p>text The text of the identifier element</p>

	<p>domain The domain which the identifier is applicable to</p> <p>project.id The ID of the project</p> <p>site.id One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the ID of the site.</p> <p>tree.id The ID of the tree</p> <p>core.id The ID of the core</p> <p>radius.id The ID of the radius</p> <p>measurement.id The ID of the measurement series</p> <p>derived.id The ID of the derived series</p> <p>project.title The title of the project</p> <p>site.title One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the title of the site.</p> <p>tree.title The title of the tree</p> <p>core.title The title of the core</p> <p>radius.title The title of the radius</p> <p>measurement.title The title of the measurement series</p> <p>derived.title The title of the derived series</p>
remark	<p>A list of remarks concerning individual measured or derived values, with some of the following items:</p> <p>measurements Remarks related to measurements with a known year. A <code>data.frame</code> with the following columns:</p> <ul style="list-style-type: none"> text The remark frame Index to a <code>data.frame</code> in <i>measurements</i> row Index to a row of the <code>data.frame</code> col Index to a column of the <code>data.frame</code> <p>undated Remarks related to measurements without a known year. A <code>data.frame</code> with the following columns:</p> <ul style="list-style-type: none"> text The remark series Index to a series in <i>undated\$data</i> idx Index to a value in the series <p>derived Remarks related to derived values. A <code>data.frame</code> with the following columns:</p> <ul style="list-style-type: none"> text The remark series Index to a series in <i>derived\$data</i> idx Index to a value in the series
laboratory	<p>A <code>data.frame</code> or a list of <code>data.frames</code> with one item per project. Each <code>data.frame</code> contains information about the research laboratories involved in the project. Columns are a subset of the following:</p> <ul style="list-style-type: none"> name Name of the laboratory acronym Acronym of the name identifier Identifier domain Domain which the identifier is applicable to

	<p>addressLine1 Address</p> <p>addressLine2 Another address line</p> <p>cityOrTown City or town</p> <p>stateProvinceRegion State, province or region</p> <p>postalCode Postal code</p> <p>country Country</p>
research	<p>A <code>data.frame</code> or a list of <code>data.frames</code> with one item per project. Each <code>data.frame</code> contains information about the systems in which the research project is registered. Columns are the following:</p> <p>identifier Identifier</p> <p>domain Domain which the identifier is applicable to</p> <p>description General description</p>
altitude	<p>A <code>data.frame</code> containing the altitude of trees. Columns are the following:</p> <p>metres The altitude in metres</p> <p>project.id The ID of the project</p> <p>site.id One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the ID of the site.</p> <p>tree.id The ID of the tree</p> <p>project.title The title of the project</p> <p>site.title One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the title of the site.</p> <p>tree.title The title of the tree</p>
preferred	<p>A <code>data.frame</code> containing links to preferred measurement series. Columns are a subset of the following:</p> <p>idRef Reference to a series in the same file</p> <p>xLink URI</p> <p>identifier Identifier of a series not necessarily in the same file</p> <p>domain Domain which the identifier is applicable to</p> <p>project.id The ID of the project</p> <p>site.id One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the ID of the site.</p> <p>tree.id The ID of the tree</p> <p>project.title The title of the project</p> <p>site.title One or more columns with this prefix, depending on the maximum depth of the <object> hierarchy. Gives the title of the site.</p> <p>tree.title The title of the tree</p>

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

ReferencesTRiDaS - The Tree Ring Data Standard, <http://www.tridas.org/>**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"))
```

Arguments

<code>fname</code>	a character vector giving the file name of the rwl file.
<code>header</code>	logical flag indicating whether the file has a header. If NULL then the function will attempt to determine if a header exists.
<code>long</code>	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.
<code>encoding</code>	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 .

Details

This reads in a standard rwl file as defined according to the standards of the ITRDB at <http://www.ncdc.noaa.gov/paleo/treeinfo.html>. Despite the standards at the ITRDB, this occasionally fails due to formatting problems.

Value

A `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

<code>x</code>	a numeric vector representing a possibly unevenly spaced time-series.
<code>t</code>	a numeric vector of times or ages corresponding to <code>x</code> . See <code>txOrdered</code> .
<code>tType</code>	a character string indicating the type of the <code>t</code> vector: either times or ages.
<code>nsim</code>	a numeric value giving the number of simulated AR1 spectra to compute.
<code>mctest</code>	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".
<code>ofac</code>	oversampling factor for Lomb-Scargle Fourier transform. A numeric value.
<code>hifac</code>	maximum frequency to analyze relative to the Nyquist frequency. A numeric value.
<code>n50</code>	number of segments. The segments overlap by about 50 percent.
<code>rhopre</code>	a numeric value giving the prescribed autocorrelation coefficient. If NULL or negative, the autocorrelation coefficient will be estimated from the data.
<code>p</code>	a numeric or <code>bigq</code> vector of significance levels for a statistical test considering the number of runs in a sequence. See 'Details'.
<code>iwin</code>	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 <code>iwin = 2</code> corresponds to the "hanning" window.

<code>txOrdered</code>	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.
<code>verbose</code>	a logical flag. If TRUE, the function prints some information while operating.
<code>seed</code>	a value to be given to <code>set.seed</code> 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.
<code>maxTime</code>	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 <code>nLimit</code> .
<code>nLimit</code>	a numeric value giving the maximum n for which <code>runcrit</code> 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 <code>maxTime</code> . If <code>nLimit</code> is very large, <code>maxTime</code> may be (greatly) exceeded while computing the aforementioned part of the exact solution.
<code>n</code>	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, 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:

<code>varx</code>	variance of x estimated from its spectrum. A numeric value.
<code>rho</code>	average autocorrelation coefficient, either estimated from the data or prescribed (<code>rho<pre></pre></code>). A numeric value.
<code>tau</code>	the time scale of an AR1 process corresponding to <code>rho</code> . A numeric value.
<code>rcnt</code>	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 <code>iwin == 0</code> ("rectangular"), <code>ofac == 1</code> and <code>n50 == 1</code> . Otherwise the test is not performed and <code>rcnt</code> is NULL.
<code>rcritlo</code>	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 <code>redfit</code> , NULL when <code>rcnt</code> is NULL.
<code>rcrithi</code>	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 <code>redfit</code> , NULL when <code>rcnt</code> is NULL.
<code>rcritexact</code>	a logical vector specifying whether each pair of <code>rcritlo</code> and <code>rcrithi</code> are exact values (TRUE) or approximated from a normal distribution (FALSE). When returned from <code>redfit</code> , NULL when <code>rcnt</code> is NULL.
<code>freq</code>	the frequencies used. A numeric vector. The other numeric vectors have the same length, i.e. one value for each frequency studied.
<code>gxx</code>	estimated spectrum of the data (t, x). A numeric vector.
<code>gxxc</code>	red noise corrected spectrum of the data. A numeric vector.
<code>grravg</code>	average AR1 spectrum over <code>nsim</code> simulations. A numeric vector.
<code>gredth</code>	theoretical AR1 spectrum. A numeric vector.
<code>corr</code>	a numeric vector containing the by-frequency correction: <code>gxxc</code> equals <code>gxx</code> divided by <code>corr</code> (or multiplied by the inverse correction). Also used for computing <code>ci80</code> , <code>ci90</code> , <code>ci95</code> and <code>ci99</code> .
<code>ci80</code>	a numeric vector containing the bias-corrected 80th percentile (by frequency) red noise spectrum. Only if <code>mctest</code> is TRUE.
<code>ci90</code>	a numeric vector containing the 90th percentile red noise spectrum.
<code>ci95</code>	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 <i>n50</i> argument. ofac value of the <i>ofac</i> argument. hifac value of the <i>hifac</i> argument. segskip the ideal, non-rounded difference between starting indices of consecutive segments. iwin value of the <i>iwin</i> argument. If a character value was used, this contains the corresponding number used internally. nsim value of the <i>nsim</i> argument. mctest value of the <i>mctest</i> argument. rhopre value of the <i>rhopre</i> argument.
vers	version of dplr. See packageVersion .
seed	value of the <i>seed</i> argument.
t	if duplicated values of <i>t</i> are given, the non-duplicated numeric time or age vector (see <i>tType</i>) is returned here. Otherwise NULL. See <i>txOrdered</i> .
x	if duplicated values of <i>t</i> 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)
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)

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 (dB)", 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()

```

```

# Second example with tree-ring data
# Note the long-term low-freq signal in the data. E.g.,
# crn.plot(cana157)

data(cana157)
yrs <- as.numeric(rownames(cana157))
x <- cana157[, 1]
redf.x <- redfit(x, nsim = 1000)

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

plot(redf.x[["freq"]], redf.x[["gxxc"]],
     ylim = range(redf.x[["ci99"]], redf.x[["gxxc"]]),
     type = "n", ylab = "Spectrum (dB)", 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)

```

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 <code>data.frame(tree = 1:ncol(rwi), core = rep(1, ncol(rwi)))</code> .
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 either "pearson", "kendall", or "spearman" which indicates the correlation coefficient is 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 <i>rwi</i> , 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 <i>rwi</i> . 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 <code>min.corr.overlap = 0</code> .

`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).

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

- Cook, E. R. and Kairiukstis, L. A. (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 M.K. Hughes, T.W. Swetnam, and H.F. Diaz (Eds.), *Dendroclimatology*(77-112), 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

[detrnd](#), [cor](#), [read.ids](#), [rwi.stats](#), [corr.rwl.seg](#)

Examples

```

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

## Not run:
library(graphics)
def.par <- par(no.readonly=TRUE)
## Plot the chronology showing a potential cutoff year based on EPS
eps.cut <- 0.92 # An arbitrary EPS cutoff for demonstration
gp.crn <- chron(gp.rwi)
## Running stats on the rwi with an window
foo <- rwi.stats.running(gp.rwi, gp.ids, window.length = 80)
yrs <- as.numeric(rownames(gp.crn))
bar <- data.frame(yrs = c(min(yrs), foo$mid.year, max(yrs)),
                 eps = c(NA, foo$eps, NA))
par(mar = c(2, 2, 2, 2), mgp = c(1.1, 0.1, 0), tcl = 0.25,
    mfcol = c(2, 1), xaxs='i')
plot(yrs, gp.crn[, 1], type = "n", xlab = "Year",
     ylab = "RWI", axes=FALSE)
cutoff <- max(bar$yrs[bar$eps < eps.cut], na.rm = TRUE)
xx <- c(500, 500, cutoff, cutoff)
yy <- c(-1, 3, 3, -1)
polygon(xx, yy, col = "grey80")
abline(h = 1, lwd = 1.5)
lines(yrs, gp.crn[, 1], col = "grey50")
lines(yrs, ffc$aps(gp.crn[, 1], nyrs = 32), col = "red", lwd = 2)
axis(1); axis(2); axis(3);
par(new = TRUE)
## Add EPS
plot(bar$yrs, bar$eps, type = "b", xlab = "", ylab = "",
     axes = FALSE, pch = 20, col = "blue")
axis(4, at = pretty(foo$eps))
mtext("EPS", side = 4, line = 1.1)
box()
## Second plot is the chronology after the cutoff only
## Chronology is rebuilt using just years after cutoff but
## that difference is essentially nil.
yr.mask <- yrs > cutoff
yrs2 <- yrs[yr.mask]
gp.crn2 <- chron(gp.rwi[yr.mask, ])
plot(yrs2, gp.crn2[, 1], type = "n",
     xlab = "Year", ylab = "RWI", axes=FALSE)
abline(h = 1, lwd = 1.5)

```

```

lines(yrs2, gp.crn2[, 1], col = "grey50")
lines(yrs2, ffcsaps(gp.crn2[, 1], nyrs = 32),
      col = "red", lwd = 2)
axis(1); axis(2); axis(3); axis(4)
box()
par(def.par)

## End(Not run)

```

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, two measures of sensitivity, the Gini coefficient, and first order autocorrelation (`"skew"`, `"sens1"`, `"sens2"`, `"gini.coef"`, `"ar1"`).

Note that that mean sensitivity is not a robust statistic that should rarely, if ever, be used (Bunn et al. 2013).

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 (zeta) as a useful statistic in dendrochronology *Dendrochronologia* 31 250–4.

Cook, E. R. and Kairiukstis, L.A. (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

```
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 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.

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.

Author(s)

Christian Zang. Patched and improved by Mikko Korpela.

References

Lough, J. M., 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)
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

`rw1` 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

```
data(co021)
seg.plot(co021, main = "Campito Mountain")
```

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 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 (zeta) as a useful statistic in dendrochronology *Dendrochronologia* 31 250–4.

See Also

[sens2](#), [rwl.stats](#)

Examples

```
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 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 (zeta) as a useful statistic in dendrochronology *Dendrochronologia* 31 250–4.

See Also

[sens1](#), [rwl.stats](#)

Examples

```
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 <i>rwl</i> 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 <i>series</i> . Defaults to <code>as.numeric(names(series))</code> . Ignored if <i>series</i> is an index to a column of <i>rwl</i> .
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).

<code>bin.floor</code>	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.
<code>n</code>	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
<code>prewhiten</code>	logical flag. If TRUE each series is whitened using ar .
<code>biweight</code>	logical flag. If TRUE then a robust mean is calculated using tbrm .
<code>floor.plus1</code>	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)
```

skel.plot	<i>Skeleton Plot</i>
-----------	----------------------

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 of less than 8 characters giving the ID for the data
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.

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. <http://www.ltrr.arizona.edu/skeletonplot/introcrossdate.htm>
- Meko, D. (2002) Tree-Ring MATLAB Toolbox. <http://www.mathworks.com/matlabcentral/>

See Also

[Devices](#), [hanning](#)

Examples

```
library(grDevices)
data(co021)
x <- co021[,33]
x.yrs <- as.numeric(rownames(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:
## Try cross-dating
y <- co021[, 11]
y.yrs <- as.numeric(rownames(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, ...)
```

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.
...	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

```
data(co021)
spag.plot(co021)
spag.plot(co021, zfac = 2)
```

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 <code>rwi.stats</code> (FALSE, the default) or <code>rwi.stats.legacy</code> (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 function 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

```
data(anos1)
anos1.ids <- read.ids(anos1, stc = c(4, 3, 1))
strip.rwl(anos1, ids = anos1.ids, verbose = TRUE)
```

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. (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. Springer.

See Also

[chron](#)

Examples

```
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])
```

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 <i>term</i> is not used. Defaults to FALSE.

Details

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 <i>idx</i>
In mode 2	A logical value
In mode 3	A character vector, same length as in <i>term</i>
In mode 4	A character vector

Author(s)

Mikko Korpela

References

TRiDaS – The Tree Ring Data Standard, <http://www.tridas.org/>

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. (July 2005) A Universally Unique Identifier (UUID) URN Namespace, RFC 4122. [Online]. Available: <http://tools.ietf.org/html/rfc4122>

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

```

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$Power,
                                       probs = seq(from=0, to=1, by=0.1)),
             add.coi = TRUE, add.sig = TRUE, x.lab = gettext("Time"),
             period.lab = gettext("Period"), crn.lab = gettext("RWI"),
             key.cols = rev(rainbow(length(wavelet.levels)-1)),
             key.lab = parse(text=paste0("\", gettext("Power"), "\"^2")),
             add.spline = FALSE, f = 0.5, nyrs = NULL,
             crn.col = "black", crn.lwd = 1, coi.col='black',
             crn.ylim = range(wave.list$y)*1.1, side.by.side = 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 ffcsaps ?
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.

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 available at <http://atoc.colorado.edu/research/wavelets/software.html>

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](#), [ffcsaps](#)

Examples

```
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi, prefix = "CAM", prewhiten = FALSE)
Years <- as.numeric(rownames(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)
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("white", "green", "blue", "red"))
```

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

```
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

```
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 <http://www.ncdc.noaa.gov/paleo/treeinfo.html>. 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:

<i>Description</i>	<i>Name</i>	<i>Class</i>	<i>Max Width</i>
Site ID	<i>site.id</i>	character	6
Site Name	<i>site.name</i>	character	52
Species Code	<i>spp.code</i>	character	4
State or Country	<i>state.country</i>	character	13
Species	<i>spp</i>	character	18
Elevation	<i>elev</i>	character	5
Latitude	<i>lat</i>	character or numeric	5
Longitude	<i>long</i>	character or numeric	5
First Year	<i>first.yr</i>	character or numeric	4
Last Year	<i>last.yr</i>	character or numeric	4
Lead Investigator	<i>lead.invs</i>	character	63
Completion Date	<i>comp.date</i>	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

```

data(ca533)
ca533.rwl <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwl, prefix = "CAM")
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.rwl, prefix = "CAM", 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.


```

        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

<code>rwl.df</code>	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.
<code>fname</code>	character vector giving the file name of the <code>rwl</code> file
<code>crn</code>	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 <code>crn.titles</code>). 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.
<code>prec</code>	optional numeric indicating the rounding precision of the output file when writing the data contained in <code>rwl.df</code> . 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.
<code>ids</code>	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: <code>data.frame(tree = 1:n.col, core = rep(1, n.col), ra</code>
<code>titles</code>	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, <code>titles</code> 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 <i>crn</i> . A single vector is interpreted as one type per data.frame in <i>crn</i> , 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 <i>crn</i> . 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 <i>crn</i> . The interpretation is the same as with <i>crn.types</i> , except that the default is to derive the titles from the column names of <i>crn</i> . 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 <i>crn</i> . The interpretation is the same as with <i>crn.types</i> , 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 <i>rwl.df</i> . 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 <i>rwl.df</i> , it is recycled to the correct length. The default is to use the information in <i>other.measuring.method</i> instead. Also, NA in any position of the vector means that the measuring method information for that series is looked up in <i>other.measuring.method</i> .
other.measuring.method	character vector giving the measuring method used to acquire each series of <i>rwl.df</i> . In contrast to <i>tridas.measuring.method</i> , these are free-form strings in English. If the vector is shorter than the number of columns in <i>rwl.df</i> , 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 <i>ids</i> . The length of the vector, however, must match the number of columns in <i>rwl.df</i> , or it is recycled to the correct length. If there are several measurements per sample, some elements of <i>sample.type</i> 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 <i>rwl.df</i> . The number of rows must match the number of columns in <i>rwl.df</i> . The columns are expected to be a subset of the following (descriptions almost directly quoted from TRiDaS specification): <ul style="list-style-type: none"> 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.

	<p>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.</p> <p>heartwood.presence Whether the outer (youngest) heartwood is present and if so whether it is complete. Category "complex presence / absence" in tridas.vocabulary.</p> <p>n.missing.heartwood Estimated number of missing heartwood rings to the pith. Non-negative integral value.</p> <p>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.</p> <p>sapwood.presence Whether the sapwood is present or not. Category "complex presence / absence".</p> <p>n.sapwood Number of sapwood rings measured. Non-negative integral value.</p> <p>last.ring.presence Last ring under the bark is present or absent. Category "presence / absence".</p> <p>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.</p> <p>n.missing.sapwood Estimated number of missing sapwood rings to the bark. Non-negative integral value.</p> <p>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.</p> <p>bark.presence Bark is present or absent. Category "presence / absence" in tridas.vocabulary.</p>
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 <i>rwl.df</i> . 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 <i>rwl.df</i> . Defaults to "ring width".
other.variable	character string. Measured variable as a free-form string. The same string is used for all of <i>rwl.df</i> . This is only used if <i>tridas.variable</i> is NA.
project.info	list containing information about the project. Elements are the following (includes quotes from the TRiDaS specification): <p>type character vector. The type(s) of the project. Defaults to "unknown".</p> <p>description character string. A description of the project. Defaults to NULL: no description.</p> <p>title character string. The title of the project. Defaults to an empty string.</p> <p>category character string. Category of research this project falls into. Defaults to an empty string.</p> <p>investigator character string. Principal investigator of this project. Defaults to an empty string.</p>

	<p>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.</p>
lab.info	<p>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:</p> <p>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</p>
research.info	<p>optional data.frame with information about the systems in which the research project is registered. Columns are the following:</p> <p>identifier Identifier domain Domain which the identifier is applicable to description General description</p>
site.info	<p>list containing information about the site (<object>). Elements are the following, and all are character strings:</p> <p>type Type of the site. Defaults to "unknown". description Description. Defaults to no description. title Title of the site. Defaults to an empty string.</p>
random.identifiers	<p>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.</p>
identifier.domain	<p>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 <i>lab.info</i>.</p>
...	<p>Unknown arguments are accepted but not used.</p>

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

TRiDaS – The Tree Ring Data Standard, <http://www.tridas.org/>

See Also

[write.rwl](#), [write.tucson](#), [write.compact](#), [write.crn](#), [read.tridas](#)

Examples

```
## 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, prefix = "CAM", 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
```

write.tucson	<i>Write Tucson Format Chronology File</i>
--------------	--

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 <http://www.ncdc.noaa.gov/paleo/treeinfo.html>. 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:

<i>Description</i>	<i>Name</i>	<i>Class</i>	<i>Max Width</i>
Site ID	<i>site.id</i>	character	5
Site Name	<i>site.name</i>	character	52
Species Code	<i>spp.code</i>	character	4
State or Country	<i>state.country</i>	character	13
Species	<i>spp</i>	character	18
Elevation	<i>elev</i>	character	5
Latitude	<i>lat</i>	character or numeric	5
Longitude	<i>long</i>	character or numeric	5
First Year	<i>first.yr</i>	character or numeric	4
Last Year	<i>last.yr</i>	character or numeric	4
Lead Investigator	<i>lead.invs</i>	character	63
Completion Date	<i>comp.date</i>	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

```
data(co021)
co021.hdr <- list(site.id = "C0021",
                 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

```

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)

```

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 <i>rwl</i> 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 <i>series</i> . Defaults to <code>as.numeric(names(series))</code> .
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 .

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.

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

[ccf](#)

Examples

```
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

<code>rw1</code>	a <code>data.frame</code> with series as columns and years as rows such as that produced by read.rw1 .
<code>series</code>	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of <code>rw1</code> is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
<code>series.yrs</code>	a numeric vector giving the years of <code>series</code> . Defaults to <code>as.numeric(names(series))</code> .
<code>win.start</code>	year to start window
<code>win.end</code>	year to end window
<code>n</code>	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
<code>prewhiten</code>	logical flag. If TRUE each series is whitened using ar .
<code>biweight</code>	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., RWD).

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.

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

[ccf](#)

Examples

```
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)
```

Index

*Topic **IO**

- read.compact, 51
- read.crn, 52
- read.fh, 53
- read.rwl, 57
- read.tridas, 58
- read.tucson, 65
- write.compact, 95
- write.crn, 96
- write.rwl, 98
- write.tridas, 99
- write.tucson, 105

*Topic **datasets**

- anos1, 4
- ca533, 8
- cana157, 9
- co021, 15
- gp.d2pith, 34
- gp.dbh, 34
- gp.po, 35
- gp.rwl, 36

*Topic **hplot**

- crn.plot, 21
- morlet, 42
- plot.rwl, 43
- seg.plot, 78
- skel.plot, 83
- spag.plot, 85
- wavelet.plot, 92
- xskel.ccf.plot, 107
- xskel.plot, 108

*Topic **htest**

- redfit, 66

*Topic **iplot**

- i.detrend, 37
- i.detrend.series, 38

*Topic **manip**

- bai.in, 5
- bai.out, 6

- ccf.series.rwl, 10
- chron, 12
- cms, 13
- combine.rwl, 15
- common.interval, 16
- corr.rwl.seg, 17
- corr.series.seg, 19
- detrend, 23
- detrend.series, 25
- fill.internal.NA, 30
- insert.ring, 39
- interseries.cor, 40
- po.to.wc, 44
- powt, 46
- rcs, 49
- series.rwl.plot, 81
- strip.rwl, 86
- wc.to.po, 94

*Topic **misc**

- read.ids, 54
- rwi.stats.running, 71
- rwl.stats, 76

*Topic **package**

- dplR-package, 3

*Topic **print**

- print.redfit, 47

*Topic **robust**

- tbrm, 87

*Topic **smooth**

- ffcsaps, 29

*Topic **ts**

- glk, 33
- hanning, 36
- redfit, 66
- sea, 77

*Topic **univar**

- gini.coef, 32
- sens1, 79
- sens2, 80

- tbrm, 87
- *Topic **utilities**
 - tridas.vocabulary, 88
 - uuid.gen, 90
- *Topic
 - strip.rwl, 86
- .Platform, 90
- .Random.seed, 91
- %dopar%, 25, 74
- anos1, 4
- approx, 31
- ar, 10–13, 18–20, 27, 41, 72, 82, 107, 109
- autoread.ids (read.ids), 54
- axis, 18
- bai.in, 5, 7
- bai.out, 5, 6, 6
- bigq, 66
- ca533, 8, 38, 54, 73
- cana157, 9
- ccf, 10, 11, 108, 109
- ccf.series.rwl, 10, 19, 21, 82
- chron, 3, 12, 14, 21–23, 50, 77, 87, 88, 96, 97, 100
- cms, 13, 50
- co021, 15
- combine.rwl, 15
- common.interval, 16
- cor, 72, 74
- cor.test, 18, 20, 41
- corr.rwl.seg, 3, 11, 17, 41, 73, 74, 82
- corr.series.seg, 11, 19, 19, 21, 82
- crn.plot, 13, 21
- delete.ring (insert.ring), 39
- detrend, 3, 12–14, 23, 28, 30, 50, 72, 74
- detrend.series, 4, 23–25, 25, 37–39
- Devices, 24, 84
- digest, 91
- dplR, 40
- dplR (dplR-package), 3
- dplR-package, 3
- ffcsaps, 22, 26, 28, 29, 31, 50, 93
- file, 52, 65
- fill.internal.NA, 30
- filter, 37
- foreach, 25, 74
- format, 48
- formula, 27
- gini.coef, 32, 76
- glk, 33
- gp.d2pith, 34
- gp.dbh, 34
- gp.po, 35
- gp.rwl, 34, 35, 36
- Grid, 108, 109
- hanning, 10, 11, 18–20, 30, 36, 41, 72, 82–84, 107, 109
- i.detrend, 37
- i.detrend.series, 38
- insert.ring, 39
- interseries.cor, 40
- lines, 85
- match.call, 69
- morlet, 42, 92, 93
- nls, 26, 27
- packageVersion, 69
- plot, 22, 49
- plot.crn (crn.plot), 21
- plot.rwl, 43
- po.to.wc, 44, 94
- pointer, 45
- powt, 46, 50
- print.data.frame, 48
- print.redfit, 47, 69
- R.version, 90
- Random, 91
- rcs, 14, 44, 47, 49, 53, 94
- read.compact, 51, 58, 65, 66, 95, 96, 100, 105
- read.crn, 9, 52, 97
- read.fh, 47, 51, 53, 58, 65, 66, 86
- read.ids, 4, 54, 73, 74
- read.rwl, 3–5, 7, 8, 10, 12, 13, 15, 16, 18, 20, 24, 30, 31, 38, 41, 44, 45, 47, 49, 51, 53, 54, 57, 57, 65, 66, 76, 77, 79, 81, 85, 86, 95, 98, 99, 105, 107, 109
- read.tridas, 51, 58, 58, 66, 89, 94, 100, 104
- read.tucson, 51, 58, 65, 65, 100, 106

redfit, [4](#), [48](#), [66](#)
runcrit (redfit), [66](#)
rwi.stats, [41](#), [57](#), [74](#), [77](#), [86](#), [87](#)
rwi.stats (rwi.stats.running), [71](#)
rwi.stats.legacy, [74](#)
rwi.stats.running, [71](#)
rwl.stats, [33](#), [41](#), [76](#), [80](#), [81](#)

save, [4](#), [8](#), [9](#), [15](#)
scale, [85](#)
sea, [77](#)
seg.plot, [16](#), [17](#), [43](#), [78](#), [85](#)
sens1, [76](#), [79](#), [81](#)
sens2, [76](#), [80](#), [80](#)
series.rwl.plot, [11](#), [19](#), [21](#), [41](#), [81](#)
skel.plot, [11](#), [19](#), [21](#), [46](#), [83](#)
spag.plot, [43](#), [79](#), [85](#)
spline, [31](#)
strip.rwl, [86](#)
summary, [76](#)
summary.rwl (rwl.stats), [76](#)
Sys.info, [90](#)

tbrm, [10–12](#), [18–20](#), [41](#), [49](#), [82](#), [87](#), [107](#), [109](#)
tridas.vocabulary, [88](#), [101](#), [102](#)

uuid.gen, [90](#), [103](#)

wavelet.plot, [42](#), [43](#), [92](#)
wc.to.po, [44](#), [94](#)
write.compact, [51](#), [95](#), [99](#), [104](#), [106](#)
write.crn, [96](#), [97](#), [99](#), [104](#), [106](#)
write.rwl, [58](#), [96](#), [98](#), [104](#), [106](#)
write.tridas, [44](#), [65](#), [89](#), [96](#), [99](#), [99](#), [106](#)
write.tucson, [66](#), [96](#), [99](#), [104](#), [105](#)

xskel.ccf.plot, [107](#)
xskel.plot, [108](#)