Package: dplR (via r-universe)

October 31, 2026

Encoding UTF-8

Type Package

Title Dendrochronology Program Library in R

Version 1.8.0

Copyright Authors and file inst/COPYRIGHTS

Depends R (>= 3.5.0)

Imports graphics, grDevices, grid, stats, utils, lattice (>= 0.13-6), Matrix (>= 1.0-3), digest (>= 0.2.3), matrixStats (>= 0.50.2), png (>= 0.1-2), R.utils (>= 1.32.1), stringi (>= 0.2-3), stringr (>= 0.4), XML (>= 2.1-0), signal, boot, lme4, lifecycle, data.table (>= 1.14.0)

Suggests Cairo (>= 1.5-0), foreach, gmp (>= 0.5-5), iterators, knitr, RColorBrewer, rmarkdown (>= 2.12), testthat (>= 0.8)

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

LazyData no

License GPL (>= 2)

VignetteBuilder knitr

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

NeedsCompilation yes

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

Maintainer Andy Bunn <bunna@wwu.edu>

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

Date/Publication 2026-10-01 07:30:09 UTC

RemoteUrl https://github.com/cran/dplR

RemoteRef HEAD

RemoteSha 8ddc5fd9f615657e20c4d41e87b9b5231092a1a6

Contents

dplR-package . . . 4
ads . . . 5
anos1 . . . 7
as.bai . . . 7
as.rwi . . . 9
as.rwl . . . 12
bai.in . . . 13
bai.out . . . 14
bakker . . . 16
ca533 . . . 18
cana157 . . . 19
caps . . . 19
ccf.series.rwl . . . 21
check.rwl . . . 24
chron . . . 26
chron.ars . . . 27
chron.ci . . . 29
chron.stabilized . . . 30
cms . . . 32
co021 . . . 34
combine.rwl . . . 35
common.interval . . . 36
corr.rwl.seg . . . 37
corr.series.seg . . . 41
detrend . . . 44
detrend.series . . . 46
dplR-defunct . . . 51
Extract.rwl . . . 51
fill.internal.NA . . . 54
gini.coef . . . 56
glk . . . 57
gp.d2pith . . . 59
gp.dbh . . . 60
gp.po . . . 60
gp.rwl . . . 61
hanning . . . 62
i.detrend . . . 63
i.detrend.series . . . 64
insert.ring . . . 65
interseries.cor . . . 66

latexDate . . . 68
latexify . . . 69
morlet . . . 71
net . . . 73
nm046 . . . 74
pass.filt . . . 75
plot.crn . . . 77
plot.crs . . . 78
plot.rwl . . . 79
po.to.wc . . . 80
pointer . . . 81
powt . . . 83
print.redfit . . . 84
print.rwl.report . . . 86
rasterPlot . . . 87
rcs . . . 90
read.compact . . . 93
read.crn . . . 94
read.fh . . . 95
read.ids . . . 96
read.rwl . . . 100
read.sheet . . . 101
read.tridas . . . 104
read.tucson . . . 111
read.tucson.legacy . . . 114
redfit . . . 115
rwi.stats.running . . . 121
rwl.check . . . 125
rwl.report . . . 128
rwl.stats . . . 130
sea . . . 131
seg.plot . . . 133
sens1 . . . 134
sens2 . . . 135
series.rwl.plot . . . 136
sgc . . . 138
skel.plot . . . 139
spag.plot . . . 141
ssf . . . 143
sss . . . 146
strip.rwl . . . 148
tbrm . . . 149
time.rwl . . . 151
treeMean . . . 152
tridas.vocabulary . . . 153
uuid.gen . . . 155
wa082 . . . 157
wavelet.plot . . . 158

wc.to.po 160
window.rwl 161
write.compact 163
write.crn 164
write.rwl 166
write.sheet 167
write.tridas 169
write.tucson 175
xdate.floater 178
xdate.report 181
xskel.ccf.plot 185
xskel.plot 187
zof.anc 189
zof.rwl 189

Index **191**

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

Description

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

Details

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

Main Functions

`read.rwl` reads rwl files

`detrend` detrends raw ring widths

`chron` builds chronologies

`corr.rwl.seg` crossdating function

Author(s)

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

References

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

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

`ads` *Age-Dependent Spline*

Description

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

Usage

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

Arguments

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

Details

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

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

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

Value

A filtered vector.

Author(s)

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

References

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

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

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

See Also

caps, detrend

Examples

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

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

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

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

anos1 *Rothenburg Tree Ring Widths*

Description

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

Usage

```
data(anos1)
```

Format

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

References

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

as.bai *Basal Area Increment (bai) Objects*

Description

Class `"bai"` holds basal area increment: the area of each ring, with one series per column and one year per row, as made by `bai.in` and `bai.out`. `as.bai` labels a `data.frame` or `matrix` of ring areas as one.

Usage

```
as.bai(x)

## S3 method for class 'bai'
x[i, j, drop]

## S3 method for class 'bai'
subset(x, subset, select, drop = FALSE, ...)

## S3 method for class 'bai'
time(x, ...)

## S3 method for class 'bai'
```

```
summary(object, ...)

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

Arguments

x, object	for as.bai, a data.frame or matrix with series as columns and years as row names; for the methods, an object of class "bai".
i, j, subset, select, drop	as for [.rwl and subset.rwl.
plot.type	"seg" for seg.plot or "spag" for spag.plot.
...	for plot, passed to the plotting function. Otherwise not used.

Details

A "bai" object has the same shape as an "rwl" object and makes the same promise: the row names are the years, consecutive and increasing. It inherits from neither "rwl" nor "rwi", so that areas, widths and indices can be told apart.

dplR's functions use the class to check what they are given. Those that want ring-width indices, such as chron and rwi.stats, take a "bai" object without a word: a mean of basal area increment by year is a BAI chronology. So do the crossdating functions, the plots, common.interval, and detrend, since fitting a curve to basal area increment is established practice. Functions that want ring widths, such as bai.in, rcs, cms and rwl.report, warn and treat the areas as widths. If the values really are widths, relabel them with as.rwl.

Two attributes travel with it. attr(x, "dplR.bai") says how the areas were made: the function (fun) and whether the distance to pith (d2pith, for bai.in) or the diameters (diam, for bai.out) were given rather than estimated from the widths. attr(x, "dplR.provenance") is the provenance record of the ring widths the areas were made from, if they had one (see read.tucson). Neither is set by as.bai, which only labels the object.

Subsetting works as for "rwl" objects ([.rwl), and window takes a span of years (window.rwl). summary gives the per-series statistics of rwl.stats, in the units of the areas.

Value

as.bai: an object of class c("bai", "data.frame"). x is returned unchanged if it already has that class. time: a numeric vector of years. summary: a data.frame, as from rwl.stats.

Author(s)

Andy Bunn

See Also

bai.in, bai.out, as.rwl, as.rwi, chron

Examples

```
library(utils)
data(gp.rwl)
data(gp.d2pith)
gp.bai <- bai.in(rwl = gp.rwl, d2pith = gp.d2pith)
class(gp.bai)
str(attr(gp.bai, "dplR.bai"))
head(summary(gp.bai))
## A BAI chronology
gp.bai.crn <- chron(gp.bai)
plot(gp.bai.crn, add.spline = TRUE, nyrs = 32)
plot(gp.bai[, 1:5])
```

`as.rwi` *Ring-Width Index (rwi) Objects*

Description

Class `"rwi"` holds ring-width indices: detrended series with one series per column and one year per row, as made by `detrend`, `rcs` and `cms`. `as.rwi` labels a `data.frame` or `matrix` of indices as one.

Usage

```
as.rwi(x)

## S3 method for class 'rwi'
x[i, j, drop]

## S3 method for class 'rwi'
subset(x, subset, select, drop = FALSE, ...)

## S3 method for class 'rwi'
time(x, ...)

## S3 method for class 'rwi'
summary(object, ids = NULL, pcrit = 0.05, ...)

## S3 method for class 'summary.rwi'
print(x, max.print = 10, ...)

## S3 method for class 'summary.rwi'
as.data.frame(x, ...)

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

Arguments

x, object	for as.rwi, a data.frame or matrix with series as columns and years as row names; for the methods, an object of class "rwi", or of class "summary.rwi" for print and as.data.frame.
ids	an optional data.frame grouping the series by tree, with columns tree and core, one row per series, as for rwi.stats (see read.ids). NULL counts each series as its own tree.
pcrit	the significance level below which a series is taken to correlate with the others.
max.print	the most series to list as not correlating.
i, j, subset, select, drop	as for [.rwl and subset.rwl.
plot.type	"spag" for spag.plot, "seg" for seg.plot, or "image" for a heat map of the indices. See Details.
...	for plot, passed to the plotting function; for plot.type = "image" this includes clip, the quantile at which the colours are clipped (default 0.99), and arguments to image. Otherwise not used.

Details

An "rwi" object has the same shape as an "rwl" object and makes the same promise: the row names are the years, consecutive and increasing. It does not inherit from "rwl", so that indices and ring widths can be told apart.

dplR's functions use the class to check what they are given. Those that want ring widths – detrend, rcs, cms, i.detrend, bai.in, bai.out, pointer, strip.rwl, ssf and rwl.report – warn when given an "rwi" object, and say what goes wrong: detrend on indices, for instance, divides out a growth curve that has already been removed. Those that want indices – chron, chron.ars, chron.stabilized, rwi.stats, rwi.stats.running and sss – warn when given an "rwl" object: rwi.stats(ca533) gives an rbar.eff of 0.350 from the widths against 0.423 from the Spline indices, and nothing about the number says it is wrong. They still take a plain data.frame or matrix without a word, and a "bai" object of basal area increment (see as.bai). The crossdating functions, the plots, common.interval, sgc and rwl.stats take either class quietly. Each warning is only a warning, and the function goes on: if the class is what is wrong – indices read from a file with read.rwl come back as class "rwl" – relabel them with as.rwi, or relabel widths with as.rwl.

Two attributes travel with it. attr(x, "dplR.detrend") is a list that says how the indices were made: the function (fun), the method and its settings, and always difference, which is TRUE when the indices are differences from the fitted curve (centred on 0) rather than ratios (centred on 1). attr(x, "dplR.provenance") is the provenance record of the ring widths the indices were made from, if they had one (see read.tucson). Neither is set by as.rwi, which only labels the object.

Subsetting works as for "rwl" objects ([.rwl): the class and both records are kept, dropping series trims the years that none of the remaining series cover, and a row subset that leaves the years out of order or with holes returns a plain data.frame, with a warning. subset and window.rwi drop series left with no values and name them in a message; [keeps them, on purpose, so that its columns stay lined up with anything indexed alongside them.

summary describes the indices as a collection. It gives how they were made (from attr(x, "dplR.detrend")), their span and common interval, the collection statistics from rwi.stats (including rbar.eff, EPS

and SNR), and for each series its first and last years, length, mean, standard deviation, first-order autocorrelation, and its correlation with the mean of the other series, with the p-value, from `interseries.cor` with its defaults. Printing it shows the collection and lists the series that do not correlate with the others at `pcrit`; `as.data.frame` gives the per-series table. The collection statistics count each series as its own tree unless `ids` is given, and the print says which. The print ends by pointing to `summary` with `ids` (when it was not given), `rwi.stats.running` (rbar and EPS through time, which usually fall off where sample depth does) and `corr.rwl.seg` (where in a series the fit breaks down). A series with no values at all is listed and left out of the collection statistics, the correlations and the common interval. With fewer than two series with values there is no collection, and the statistics and correlations are `NULL` and `NA`.

`plot` draws, by default, `spag.plot`, which draws indices around 1 (or 0 for differences) and not around each series' mean, so the grey line under each series is the value the indices should sit at. `plot.type = "image"` draws each index as a coloured cell, with years across and series down, earliest-starting at the bottom. Colours are brown below 1 (or 0) and green above it. Each side is scaled separately and clipped at the `clip` quantile of its departures, since ratio indices cannot fall below 0 but can run well above 2; the key gives the clips. A run of one colour at the start or end of a series is a growth trend the detrending did not remove, and a vertical stripe is a year the series agree on.

Arithmetic on an `"rwi"` object (e.g. `x + 1`) returns a plain `data.frame`, as it does for any `data.frame` subclass. Use `as.rwi` on the result if it is still a set of indices.

Value

`as.rwi`: an object of class `c("rwi", "data.frame")`. `x` is returned unchanged if it already has that class. `time`: a `numeric` vector of years. `summary`: an object of class `"summary.rwi"`, a list with `how` (the `dplR.detrend` record), `n.series`, `first`, `last`, `common` (the first and last years in which every series has a value, or `NA`), `stats` (from `rwi.stats`), `ids.given`, `empty` (the names of series with no values), `series` (the per-series `data.frame`) and `pcrit`.

Author(s)

Andy Bunn

See Also

`detrend`, `rcs`, `cms`, `as.rwl`, `chron`, `rwi.stats`

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(ca533, method = "Spline")
class(ca533.rwi)
str(attr(ca533.rwi, "dplR.detrend"))
summary(ca533.rwi)
## Group cores by tree, from the series IDs
summary(ca533.rwi, ids = autoread.ids(ca533))
head(as.data.frame(summary(ca533.rwi)))
plot(ca533.rwi[, 1:5])
data(co021)
```

```
## The growth trend "Mean" leaves in: a run of green at the start of
## nearly every series.
plot(detrend(co021, method = "Mean"), plot.type = "image")
```

`as.rwl` *as.rwl*

Description

Attempts to turn its argument into a rwl object.

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

Examples

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

```
plot(foo, plot.type="spag")
summary(foo)
```

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

Description

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

Usage

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

Arguments

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

Details

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

Value

An object of class `c("bai", "data.frame")` containing the ring areas for each series with the column names, row names and dimensions of *rwl*. It carries the provenance record of *rwl*, if it has one, and records how it was made in `attr(x, "dplR.bai")`: the function (`fun`) and whether *d2pith* was given (`d2pith`). See `as.bai` for which functions take it.

Note

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

Author(s)

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

References

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

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

See Also

bai.out, as.bai

Examples

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

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

## Identical
head(bar)
head(foo)

## Use gp data
data(gp.rwl)
data(gp.d2pith)
foo <- bai.in(rwl = gp.rwl, d2pith = gp.d2pith)
foo.crn <- chron(foo)
yrs <- time(foo.crn)
plot(yrs, foo.crn[, 1], type = "n",
     xlab = "Year", ylab = expression(mm^2))
lines(yrs, foo.crn[, 1], col = "grey", lty = "dashed")
lines(yrs, caps(foo.crn[, 1], nyrs = 32), col = "red", lwd = 2)
```

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

An object of class c("bai", "data.frame") containing the ring areas for each series with the column names, row names and dimensions of *rwl*. It carries the provenance record of *rwl*, if it has one, and records how it was made in attr(x, "dplR.bai"): the function (fun) and whether *diam* was given (diam). See as.bai for which functions take it.

Note

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

Author(s)

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

References

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

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

See Also

bai.in, as.bai

Examples

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

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

## Identical
head(bar)
head(foo)

## End(Not run)
## Use gp data
data(gp.rwl)
data(gp.dbh)
## dbh (minus the bark) from cm to mm
gp.dbh2 <- gp.dbh[, 1:2]
gp.dbh2[, 2] <- (gp.dbh[, 2] - gp.dbh[, 3]) * 10
bar <- bai.out(rwl = gp.rwl, diam = gp.dbh2)
bar.crn <- chron(bar)
yrs <- time(bar.crn)
plot(yrs, bar.crn[, 1], type = "n",
     xlab = "Year", ylab = expression(mm^2))
lines(yrs, bar.crn[, 1], col = "grey", lty = "dashed")
lines(yrs, caps(bar.crn[, 1], nyrs = 32), col = "red", lwd = 2)
```

---

bakker *Basal Area Increment (Bakker)*

Description

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

Usage

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

Arguments

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

Details

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

Value

A list containing the following objects:

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

Author(s)

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

References

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

Examples

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

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

ca533 *Campito Mountain Tree Ring Widths*

Description

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

Usage

```
data(ca533)
```

Format

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

Source

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

References

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

Examples

```
library(utils)
data(ca533)

## Where the data came from. read.tucson() records what it saw and attaches
## it to what it returns; this data set carries that record.
prov <- attr(ca533, "dplR.provenance")
prov$file          # the archived file it was read from
prov$precision     # the precision each series was measured at
prov$gaps          # interior gaps, if any -- this file has none
prov$events        # problems found while reading -- none here either

## rwl.report() puts the file and precision at the head of its report
rwl.report(ca533)
```

cana157 *Twisted Tree Heartrot Hill Standard Chronology*

Description

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

Usage

```
data(cana157)
```

Format

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

Source

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

References

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

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

Description

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

Usage

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

Arguments

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

Details

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

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

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

Note: *nyrs* is passed to the Fortran subroutine as an integer, so a fractional *nyrs* is truncated. This matters when *nyrs* is given as a proportion of series length, which is rarely a whole number. The effect on the fitted curve is small but not zero; see the "Mathematical Details" vignette for a worked comparison.

Note: *caps* stops if there are any NA values in *y*. A smoothing spline through a series with missing values is not defined, and the caller has to decide what to do about the gap rather than have one chosen silently. Before version 1.8.0 such a call returned a vector of NA, which looked like an answer and was not. See examples, and `fill.internal.NA`.

Value

A filtered vector.

Note

The mathematical relationship between *nyrs*, *f* and the spline's smoothing parameter, together with an empirical check of the frequency response and a demonstration that *caps* reproduces the deprecated *ffcsaps*, is given in the vignette: `vignette("math-dplR", package = "dplR")`.

Author(s)

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

References

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

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

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

See Also

ads

Examples

```
library(graphics)
library(utils)

## Use first series from the Mesa Verde data set
data(co021)
series <- co021[, 1]
series <- series[!is.na(series)]
plot(series, type = "l", ylab = "Ring Width (mm)", col = "grey")
lines(caps(series, nyrs = 10), col = "red", lwd = 2)
lines(caps(series, nyrs = 100), col = "green", lwd = 2)
# The default, a 32-year spline since version 1.7.8
lines(caps(series), col = "blue", lwd = 2)
legend("topright",
       c("Series", "nyrs=10", "nyrs=100", "Default nyrs (32)"),
       fill=c("grey", "red", "green", "blue"))


## caps() needs a series with no missing values. Note that the example
## above drops the NA padding before fitting. A gap inside the measured
## span has to be dealt with deliberately: drop the series, or close the
## gap with fill.internal.NA().
y <- c(NA, NA, rnorm(100))
try(caps(y))
## fit over the measured values instead
head(caps(y[!is.na(y)]))
```

---

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

Description

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

Usage

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

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)). Ignored if *series* is an index to a column of *rwl*.
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
bin.floor	a non-negative integral value giving the base for locating the first segment (e.g., 1600, 1700, 1800 AD). Typically 0, 10, 50, 100, etc.
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
nyrs	NULL or a number greater than zero. If not NULL, each series is divided by a smoothing spline with this rigidity (see caps) to remove low-frequency variation, as detrend does with method = "Spline". A value of 1 or less is taken as a proportion of each series' length. Unlike the hanning filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
prewhiten	logical flag. If TRUE each series is whitened using ar.
ar.order.max	NULL or a positive integer giving the maximum order of the ar model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If NULL, the order is chosen by AIC up to the ar default, which on long series can be 20 or more. Requires *prewhiten* = TRUE.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
pcrit	a number between 0 and 1 giving the critical value for the correlation test.
lag.max	an integral value giving the maximum lag at which to calculate the ccf.
make.plot	logical flag indicating whether to make a plot.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
series.x	logical flag indicating whether to make the series the x argument to ccf. See Details.
...	other arguments passed to plot.

Details

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

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

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

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

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

See Also

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

Examples

```
library(utils)
data(co021)
dat <- co021
## Create a missing ring: delete the 1500 ring of series 641143 and
## drop the original from the master.  Dated from the bark, every ring
## before 1500 now sits one year late.
```

```
flagged <- dat$"641143"
names(flagged) <- rownames(dat)
flagged <- delete.ring(flagged, year = 1500)
dat$"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)
```

`check.rwl` *Check and Validate an rwl Object*

Description

Checks that an object is a valid `rwl` and, if not, attempts coercion via `as.rwl`. Warns about internal `NA` values found within any series.

Usage

```
check.rwl(rwl, why = NULL, bai.ok = FALSE)
```

Arguments

`rwl`	an `rwl` object such as that produced by `read.rwl`. Also accepts a `data.frame` or `matrix` with series as columns and years as row names, which will be coerced to `rwl` with a warning.
`why`	`NULL` or a character string, added to the warning given when `rwl` is an `"rwi"` object, to say what goes wrong if indices are used as ring widths. `NULL` gives a general sentence.
`bai.ok`	logical. If `TRUE`, a `"bai"` object of basal area increment (see `as.bai`) is taken without a warning. `detrend` uses this.

Details

`check.rwl` is for functions that want ring widths. If `rwl` is an `"rwi"` object of ring-width indices (see `as.rwi`), a warning names the calling function, says that it wants widths, adds `why`, and suggests `as.rwl` if the values really are widths. The object is then relabelled as class `"rwl"` and checked as

below. A "bai" object of basal area increment (see as.bai) is treated the same way, unless bai.ok is TRUE, when it is relabelled without a warning.

If rwl is not already of class "rwl", coercion is attempted using as.rwl. If coercion succeeds, a warning is issued. If coercion fails (e.g., because row names are not consecutive integers, columns are not numeric, or the input is not a data.frame or matrix), the function stops with a single informative message that describes both the class problem and the reason coercion failed.

After class validation, each series is checked for internal NA values, defined as NA values that are sandwiched between non-NA values within a series (as opposed to leading or trailing NAs, which are standard in rwl objects). If any are found, a warning names the affected series and suggests fill.internal.NA.

This function is called at the top of the dplR functions that want ring widths, providing uniform validation across the package. Functions that take widths or indices alike, and those that want indices, use internal checks of their own (see as.rwi).

Value

An object of class c("rwl", "data.frame") with series in columns and years as row names. The series IDs are the column names. The returned object is identical to the input when the input is already a valid rwl.

Author(s)

Andy Bunn

See Also

as.rwl, as.rwi, read.rwl, fill.internal.NA

Examples

```
library(utils)
data(ca533)

## A proper rwl passes silently and is returned unchanged
ca533.checked <- check.rwl(ca533)
identical(ca533, ca533.checked)

## A plain data.frame is coerced with a warning.
## suppressWarnings() is used here to keep R CMD check clean since the
## warning is expected. Remove it to see the coercion warning.
ca533.df <- ca533
class(ca533.df) <- "data.frame"
ca533.rwl <- suppressWarnings(check.rwl(ca533.df))
#ca533.rwl <- check.rwl(ca533.df)
class(ca533.rwl)
```

chron *Build Mean Value Chronology*

Description

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

Usage

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

Arguments

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

Details

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

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

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

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

See Also

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

Examples

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

`chron.ars` *Build ARSTAN Chronology*

Description

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

Usage

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

Arguments

`rwi`	a `data.frame` of (usually detrended) ring widths with `rownames(`*rwi*`)` containing years and `colnames(rwi)` containing each series ID such as produced by `detrend`
`biweight`	`logical` flag. If TRUE then a robust mean is calculated using `tbrm`.
`maxLag`	an `integer` giving the maximum lag to consider in the pooled AR modeling. If the AIC does not reach a local minimum within this range when using `firstAICmin` = TRUE, the function stops with an error. Increase `maxLag` if this occurs.
`firstAICmin`	`logical` flag. If TRUE the final AR order is selected using the first local AIC minimum; otherwise the order is selected by the overall minimum across all lags up to `maxLag`. See Details.
`verbose`	`logical` flag. If TRUE the function prints information from the AR modeling to the screen.
`prewhitenMethod`	a `character` vector to determine the AR model fitting. See details below. Possible values are either `"ar.yw"` or `"arima.CSS-ML"`. Can be abbreviated. Defaults to `"ar.yw"`.

Details

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

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

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

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

The order of the AR model is selected from the pooled AR coefficients by AIC using either the first local AIC minimum (`firstAICmin = TRUE`, the default) or the overall minimum across all lags up to `maxLag` (`firstAICmin = FALSE`). When `firstAICmin = TRUE` and the AIC decreases monotonically across all tested lags without reaching a local minimum, the function stops with an error. This can occur when `maxLag` is too small for the data or when the series have persistence structures that are not well-suited to this approach (e.g., non-stationary or very long-memory series). In such cases, consider increasing `maxLag` or reviewing the data preparation steps described in Cook (1985). When `firstAICmin = FALSE` the function always selects an order and does not stop in this way.

If the selected AR order is zero (no detectable common autocorrelation among the series), no prewhitening or post-reddening is applied and all three returned chronologies are identical to the standard chronology.

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

Value

A `data.frame` of class `"crn"` with the following columns:

`std`	the standard chronology: the (biweight) mean ring-width index across all series for each year.
`res`	the residual chronology: the standard chronology after multivariate autoregressive prewhitening to remove common persistence.
`ars`	the ARSTAN chronology: the residual chronology with the pooled autoregression reintroduced. When the selected AR order is zero, `std`, `res`, and `ars` are identical.
`samp.depth`	the number of series with non-missing values in each year.

Row names are the years taken from `rownames(rwi)`.

Author(s)

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

References

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

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

See Also

chron, crn.plot, ar, arima

Examples

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

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Andy Bunn.

See Also

`read.rwl`, `detrend`, `boot`, `boot.ci`

Examples

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

---

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

Description

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

Usage

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

Arguments

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

Details

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

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

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

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

Value

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

Author(s)

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

References

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

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

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

See Also

chron

Examples

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

---

cms *C-Method Standardization*

Description

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

Usage

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

Arguments

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

Details

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

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

A series with no values at all is dropped, with a message naming it, and the rest are standardized as if it had never been there. *po* must still have a row for every series in *rwl*, empty or not.

See the reference below for further details.

Value

An object of class `c("rwi", "data.frame")` (see `as.rwi`) containing the dimensionless and detrended ring-width indices with column names, row names and dimensions of *rwl* if *c.hat.t* is `FALSE` and *c.hat.i* is `FALSE`.

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

Note

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

Author(s)

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

References

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

See Also

`detrend`, `chron`, `rcs`

Examples

```
library(graphics)
library(utils)
data(gp.rwl)
data(gp.po)
gp.rwi <- cms(rwl = gp.rwl, po = gp.po)
gp.crn <- chron(gp.rwi)
```

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

---

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

Description

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

Usage

```
data(co021)
```

Format

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

Source

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

References

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

Examples

```
library(utils)
data(co021)

## Where the data came from. read.tucson() records what it saw and attaches
## it to what it returns; this data set carries that record.
prov <- attr(co021, "dplR.provenance")
prov$file          # the archived file it was read from
prov$precision     # the precision each series was measured at
prov$gaps          # interior gaps, if any -- this file has none
prov$events        # problems found while reading -- none here either
```

```
## rwl.report() puts the file and precision at the head of its report
rwl.report(co021)
```

`combine.rwl` *Combine Tree-Ring Data Sets*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Christian Zang. Patched by Mikko Korpela.

Examples

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

`common.interval` *Common Interval*

Description

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

Usage

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

Arguments

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

Details

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

Series with no values are left out. A single series is its own common interval, and its measured years are returned. If there are two or more series and no year in which any two of them overlap, or no series has any values, there is no common interval and that is an error.

Value

A `data.frame` with `colnames(x)` and `rownames(x)`, with no missing values. It has the class of *rwl*: ring-width indices (class `"rwi"`, see `as.rwi`) come back as indices. If the years kept are not consecutive, which `type = "years"` can give when a series has an interior gap, the result is a plain `data.frame`, with a warning (see `[.rwl`).

Author(s)

Filipe Campelo, Andy Bunn and Mikko Korpela

See Also

`seg.plot`

Examples

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

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

---

corr.rwl.seg *Compute Correlations between Series*

Description

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

Usage

```
corr.rwl.seg(rwl, seg.length = 50, bin.floor = 100, n = NULL,
             nyrs = NULL, prewhiten = TRUE, ar.order.max = NULL,
             pcrit = 0.05, biweight = TRUE,
             method = c("spearman", "pearson","kendall"),
             make.plot = TRUE, label.cex = 1, floor.plus1 = FALSE,
             master = NULL, lag.max = 0,
             master.yrs = as.numeric(if (is.null(dim(master))) {
                             names(master)
                         } else {
                             rownames(master)
                         }),
             ...)
```

Arguments

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

n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
nyrs	NULL or a number greater than zero. If not NULL, each series is divided by a smoothing spline with this rigidity (see caps) to remove low-frequency variation, as detrend does with method = "Spline". A value of 1 or less is taken as a proportion of each series' length. Unlike the hanning filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
prewhiten	logical flag. If TRUE each series is whitened using ar.
ar.order.max	NULL or a positive integer giving the maximum order of the ar model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If NULL, the order is chosen by AIC up to the ar default, which on long series can be 20 or more. Requires *prewhiten* = TRUE.
pcrit	a number between 0 and 1 giving the critical value for the correlation test.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
method	Can be either "pearson", "kendall", or "spearman" which indicates the correlation coefficient to be used. Defaults to "spearman". See cor.test.
make.plot	logical flag indicating whether to make a plot.
label.cex	numeric scalar for the series labels on the plot. Passed to *axis.cex* in axis.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
master	NULL, a numeric vector or a matrix-like object of numeric values, including a data.frame. If NULL, a number of master chronologies, one for each series in *rwl*, is built from *rwl* using the leave-one-out principle. If a vector, the function uses this as the master chronology. If a matrix or data.frame, this object is used for building the master chronology (no leave-one-out).
lag.max	a non-negative whole number less than *seg.length*. Each segment is also correlated against the master shifted by every lag from -*lag.max* to *lag.max* years, and the best position is returned in *best.lag* and *best.rho*. The default, 0, tests the dated position only, as before.
master.yrs	a numeric vector giving the years of *series*. Defaults to names or rownames of *master* coerced to numeric type.
...	other arguments passed to plot.

Details

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

Correlations are done for each segment of the series where segments are lagged by half the segment length (e.g., 100-year segments would be overlapped by 50-years). The first segment is placed according to *bin.floor*. The minimum bin year is calculated as ceiling(*min.yr*/*bin.floor*)**bin.floor* where *min.yr* is the first year in either the *rwl* object or the user-specified *master* chronology,

whichever is smaller. For example if the first year is 626 and *bin.floor* is 100 then the first bin would start in 700. If *bin.floor* is 10 then the first bin would start in 630.

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

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

As an alternative to the `hanning` filter, *nyrs* divides each series by a smoothing spline, which removes no years from the ends of the series. COFECHA uses a 32-year spline. The spline alone does not usually save years at the start of a series, because the `ar` model is then fitted to a high-pass filtered series, and AIC tends to choose a high order for it. Setting *ar.order.max* to a small value (e.g., 3) limits the years lost to prewhitening, and after a spline a low-order model loses little of the fit.

With neither filter, each series is divided by its mean before the master is built, so every series counts equally. Data that can be negative, such as indices from `detrend` with *difference* = `TRUE`, log widths or isotope values, have the mean subtracted instead, since dividing by a negative mean would flip the series. The *n* and *nyrs* filters divide by a smooth curve, so they refuse such data. The same applies to the other crossdating functions.

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.

A segment that correlates well where it is dated can still correlate better somewhere else. With *lag.max* greater than 0, each segment is also correlated against the master at every shift from `-`*lag.max* to *lag.max* years. Each year of the segment is paired with the master value *k* years away, so the correlation at lag *k* is that of `series[t]` with `master[t + k]`. The sign follows `ccf.series.rwl` with *series.x* = `FALSE`: a negative lag means missing rings in the series, a positive lag means false (extra) rings. The dated position wins ties.

This gives the two kinds of flag COFECHA reports. An 'A' segment correlates below the critical value, but its dated position is still the best one tested: it is weak, not misdated. A 'B' segment correlates better at some other position, whether or not it is significant as dated. The letters are not stored; they follow from the returned matrices as `best.lag != 0` for B and `best.lag == 0 & p.val >= pcrit` for A (see Examples). When *lag.max* is greater than 0, the plot draws B segments in purple and leaves red for the rest of the segments below the critical value.

A B flag is a hypothesis about the dating, not a verdict. A one-year shift that gains 0.05 in correlation is noise as often as it is a missing ring, which is why the size of the gain is returned (`best.rho - spearman.rho`) alongside the lag, so that it can be thresholded. Check any B segment against the wood, for example with `ccf.series.rwl` and `xskel.ccf.plot`.

Read the lags along a series rather than one bin at a time. Dating runs from the bark inward, so a missing ring moves every ring before it one year late: every complete bin before the missing ring has `best.lag` of -1 and every bin after it has 0. A false ring does the same with +1. The error therefore lies where the lag changes, not in the bin with the largest gain. The bin that straddles it holds some shifted and some unshifted years, and may show either lag with a small gain. A run of -1 with correctly dated bins on both sides is a different fault: the ring count is right, but a ring is missing at the later end of the run and there is an extra ring at the earlier end. A locally absent ring entered as a zero in too early a year looks exactly like this. A test on the whole series, such as `RWL_DATING_LAG` in `rwl.check`, usually cannot see it, because most of the series is dated correctly. The Examples show both patterns.

The shifted correlations follow the same complete-overlap rule as the dated one: a lag is tested only if the master has a value for every year of the shifted window. Near the start and end of the record (and of the master), the lags that run off the record cannot be tested, and a segment that lies within *lag.max* years of an end is only searched in one direction. A segment that is not complete at its dated position is not tested at all. Floating series and series with dating errors near their ends sit exactly where the test is blind, so the absence of a flag there says little. Worse, a segment misdated in the direction that runs off the record can still be flagged B, at whichever of the remaining lags happens to correlate best, so the lag reported for a B near either end can be noise even when the segment really is misdated.

Value

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

Matrices *best.lag* and *best.rho* have the same shape and names as *spearman.rho*. *best.lag* gives, for each series and bin, the lag (in years) at which the segment correlates best with the master, and *best.rho* gives that correlation. Both are `NA` where *spearman.rho* is. With *lag.max* = `0`, *best.lag* is 0 and *best.rho* equals *spearman.rho*. *lag.max* is from the original call. Flags in *flags* depend on the p-value only and are the same whatever *lag.max* is.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

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

Examples

```
library(utils)
data(co021)
crs <- corr.rwl.seg(co021, seg.length = 100, label.cex = 1.25)
```

```
names(crs)
## Average correlation and p-value for the first few series
head(crs$overall)
## Average correlation for each bin
crs$avg.seg.rho

## Plant a missing ring and follow it through the bins: delete the
## 1500 ring of series 641143, the same fault as in the examples for
## ccf.series.rwl(), corr.series.seg() and xskel.ccf.plot().  Dated
## from the bark, every ring before 1500 now sits one year late.
dat <- co021
x <- dat$"641143"
names(x) <- rownames(dat)
dat$"641143" <- delete.ring(x, year = 1500)
crs <- corr.rwl.seg(dat, lag.max = 5)
## -1 in every complete bin before 1500 (drawn in purple), 0 from 1500
## on, and a large gain in each shifted bin
ok <- !is.na(crs$best.lag["641143", ])
rbind(lag = crs$best.lag["641143", ok],
      rho = round(crs$spearman.rho["641143", ok], 2),
      best.rho = round(crs$best.rho["641143", ok], 2))

## The same missing ring with a false ring inserted at 1400.  The ring
## count is right again, and only the rings between the two errors are
## one year late.
dat$"641143" <- insert.ring(delete.ring(x, year = 1500), year = 1400)
crs <- corr.rwl.seg(dat, lag.max = 5, make.plot = FALSE)
## -1 between the two errors only, 0 on both sides
crs$best.lag["641143", ok]

## COFECHA-style A and B flags for every series
flag <- ifelse(crs$best.lag != 0, "B",
               ifelse(crs$p.val >= crs$pcrit, "A", ""))
flag[is.na(flag)] <- ""
## B segments with their lag and the gain in correlation
idx <- which(flag == "B", arr.ind = TRUE)
data.frame(series = rownames(flag)[idx[, 1]],
           bin = colnames(flag)[idx[, 2]],
           lag = crs$best.lag[idx],
           rho = round(crs$spearman.rho[idx], 3),
           best.rho = round(crs$best.rho[idx], 3))
```

`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,
                nyrs = NULL, prewhiten = TRUE, ar.order.max = NULL,
                biweight = TRUE,
                method = c("spearman", "pearson","kendall"),
                pcrit = 0.05,
                make.plot = TRUE, floor.plus1 = FALSE, ...)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)). Ignored if *series* is an index to a column of *rwl*.
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
bin.floor	a non-negative integral value giving the base for locating the first segment (e.g., 1600, 1700, 1800 AD). Typically 0, 10, 50, 100, etc.
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
nyrs	NULL or a number greater than zero. If not NULL, each series is divided by a smoothing spline with this rigidity (see caps) to remove low-frequency variation, as detrend does with method = "Spline". A value of 1 or less is taken as a proportion of each series' length. Unlike the hanning filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
prewhiten	logical flag. If TRUE each series is whitened using ar.
ar.order.max	NULL or a positive integer giving the maximum order of the ar model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If NULL, the order is chosen by AIC up to the ar default, which on long series can be 20 or more. Requires *prewhiten* = TRUE.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
method	Can be either "pearson", "kendall", or "spearman" which indicates the correlation coefficient to be used. Defaults to "spearman". See cor.test.
pcrit	a number between 0 and 1 giving the critical value for the correlation test.
make.plot	logical flag indicating whether to make a plot.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).
...	other arguments passed to plot.

Details

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

Value

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

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

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

Examples

```
library(utils)
data(co021)
dat <- co021
## Create a missing ring: delete the 1500 ring of series 641143 and
## drop the original from the master.  Dated from the bark, every ring
## before 1500 now sits one year late.
flagged <- dat$"641143"
names(flagged) <- rownames(dat)
flagged <- delete.ring(flagged, year = 1500)
dat$"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)
```

detrend *Detrend Multiple Ring-Width Series Simultaneously*

Description

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

Usage

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

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`
`y.name`	a `character` vector of `length(ncol(`*rwl*`))` that gives the ID of each series. Defaults to the column names of *rwl*.
`make.plot`	a `logical` flag. Makes plots of the raw data and detrended data if `TRUE`. See details below.
`method`	a `character` vector to determine the detrending methods. See details below. Possible values are all subsets of `c("Spline", "ModNegExp", "Mean", "Ar", "Friedman", "ModHugershoff", "AgeDepSpline")`. Defaults to `"Spline"`. Before dplR 1.8.0 the default was all the available methods.
`nyrs`	a number giving the rigidity of the smoothing spline, defaults to 0.67 of series length if *nyrs* is `NULL`.
`f`	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
`pos.slope`	a `logical` flag. Will allow for a positive slope to be used in method `"ModNegExp"` and `"ModHugershoff"`. If `FALSE` the line will be horizontal.
`constrain.nls`	a `character` string which controls the constraints of the `"ModNegExp"` model and the `"ModHugershoff"`. See `detrend.series` for further details.
`verbose`	`logical`. Write out details?
`return.info`	a `logical` flag. If `TRUE`, details about models and data will be added to the return value. See 'Value'.
`wt`	a `numeric` vector of case weights for method `"Friedman"`. The default means equals weights.
`span`	a `numeric` value controlling method `"Friedman"`, or `"cv"` (default) for automatic choice by cross-validation. See `supsmu`.

bass	a numeric value controlling the smoothness of the fitted curve in method "Friedman". See supsmu.
difference	a logical flag. Compute residuals by subtraction if TRUE, otherwise use division.

Details

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

A series with no values at all is dropped, with a message naming it, and the rest are detrended as if it had never been there; its name is dropped from *y.name* too. x[rows,] and head can leave such a series (see [.rwl). If no series has any values, that is an error. A series with NA inside its measured span stops the call, with a message naming the series; see fill.internal.NA.

Value

If one detrending method is used, an object of class c("rwi", "data.frame") containing the dimensionless detrended ring widths with the column names, row names and dimensions of *rwl*, less any series with no values. It records how it was made in attr(x, "dplR.detrend") and carries the provenance record of *rwl*, if it has one; see as.rwi. If more methods are used, a list with ncol(*rwl*) elements each containing a data.frame with the detrended ring widths in each column.

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

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

Note

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

Author(s)

Andy Bunn. Improved by Mikko Korpela.

See Also

detrend.series, as.rwi

Examples

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

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

## Not run:
library(grDevices)
## Detrend using all methods. Returns a list
ca533.rwi <- detrend(rwl = ca533,
                     method = c("Spline", "ModNegExp", "Mean", "Ar",
                                "Friedman", "ModHugershoff", "AgeDepSpline"))
## 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 = "Spline",
               nyrs = NULL, f = 0.5, pos.slope = FALSE,
               constrain.nls = c("never", "when.fail", "always"),
               verbose = FALSE, return.info = FALSE,
               wt, span = "cv", bass = 0, difference = FALSE)
```

Arguments

y	a numeric vector. Usually a tree-ring series.
y.name	an optional character vector to name the series for plotting purposes.
make.plot	a logical flag. Makes plots of the raw data and detrended data if TRUE.
method	a character vector to determine the detrending methods. See details below. Possible values are all subsets of c("Spline", "ModNegExp", "Mean", "Ar", "Friedman", "ModHugershoff", "AgeDepSpline"). Defaults to "Spline". Before dplR 1.8.0 the default was all the available methods.
nyrs	a number controlling the smoothness of the fitted curve in methods. See Details.
f	a number between 0 and 1 giving the frequency response or wavelength cutoff in method "Spline". Defaults to 0.5. See caps.
pos.slope	a logical flag. Will allow for a positive slope to be used in method "ModNegExp" or "AgeDepSpline". If FALSE the line will be horizontal.
constrain.nls	a character string which controls the constraints of the "ModNegExp" model and and the "ModHugershoff" model which are fit using nonlinear least-squares via nls. The value is an answer to the question: When should the parameters of the nls function be constrained? The options are "never": do not constrain (the default), "when.fail": only compute the constrained solution if the unconstrained fit contains other than positive values, and "always": return the constrained solution, even if the unconstrained one would have been valid. See 'Details'.
verbose	a logical flag. Write out details to the screen?
return.info	a logical flag. If TRUE, details about models and data will be added to the return value. See 'Value'.
wt	a numeric vector of case weights for method "Friedman". The default means equals weights.
span	a numeric value controlling method "Friedman", or "cv" (default) for automatic choice by cross-validation. See supsmu.
bass	a numeric value controlling the smoothness of the fitted curve in method "Friedman". See supsmu.
difference	a logical flag. Compute residuals by subtraction if TRUE, otherwise use division.

Details

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

A curve that is divided by must be positive, so when difference is FALSE a fit that is not all positive is replaced by a straight line or the mean (or, for "Ar", its negative values are set to zero)

with a warning. Subtraction has no such need, so with `difference = TRUE` the fits are used as they are, zeros are left alone, and `y` may be negative, as log widths or isotope values are.

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

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

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

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

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

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

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

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

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

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

These methods are chosen because they are commonly used in dendrochronology. There is a rich literature on detrending and many researchers are particularly skeptical of the use of the classic

nonlinear model of biological growth ($f(t) = ae^{bt} + k$) for detrending. It is, of course, up to the user to determine the best detrending method for their data.

Note that the user receives a warning if a series has negative values in the fitted curve. This happens fairly commonly with with the 'Ar' method on high-order data. It happens less often with method 'Spline' but isn't unheard of (see 'Examples'). If this happens, users should look carefully at their data before continuing. Automating detrending and not evaluating each series individually is folly. Remember, frustration over detrending is the number one cause of dendros going to live as hermits in the tallgrass prairie, where there are no trees to worry about.

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

Value

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

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

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

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

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

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

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

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

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

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

formula The `"ModNegExp"` `formula`.

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

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

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

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

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

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

Author(s)

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

References

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

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

See Also

detrend

Examples

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

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

# note the difference between ModNegExp and ModHugershoff at the
# start of the series. Also notice how curves, etc. are returned
# via return.info
data(co021)
series <- co021[, 4]
```

```
names(series) <- rownames(co021)
series.rwi <- detrend.series(y = series, y.name = names(co021)[4],
                             method=c("ModNegExp", "ModHugershoff"),
                             verbose = TRUE, return.info = TRUE,
                             make.plot = TRUE)

# A dirty dog.
# In the case of method=="Spline" the function carries-on
# and applies method=="Mean" as an alternative.
data(nm046)
series <- nm046[,8]
names(series) <- rownames(nm046)
series.rwi <- detrend.series(y = series, y.name = names(nm046)[8],
                             method="Spline",
                             make.plot = FALSE)
```

`dplR-defunct` *Defunct functions in dplR*

Description

These are functions no longer included in dplR

Details

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

`Extract.rwl` *Subset an rwl Object*

Description

Subset a `rwl` object by series (columns) and years (rows), keeping the class and the provenance record of the data.

Usage

```
## S3 method for class 'rwl'
x[i, j, drop]

## S3 method for class 'rwl'
subset(x, subset, select, drop = FALSE, ...)
```

Arguments

x	an object of class "rwl" with series as columns and years as rows, such as that produced by read.rwl.
i, j	indices for the rows (years) and columns (series), used exactly as in [.data.frame: numbers, negative numbers, logicals, or names. The year of a row is its name, so x["1900",] selects a year and x[time(x) %in% 1800:1900,] selects a window of them.
drop	logical. As in [.data.frame: if the result has one column and drop is TRUE, a numeric vector is returned instead of a one-series rwl object. Note that subset defaults to drop = FALSE, as subset.data.frame does.
subset, select	for the subset method, expressions naming the years (rows) to keep and the series (columns) to keep, used exactly as in subset.data.frame.
...	Not used.

Details

Subsetting an rwl object works as it does for a data.frame, with i indexing years and j indexing series. Four things are different.

Dropping series returns the years the remaining series cover. A collection is ragged, and the years at the ends of an rwl object usually belong to a few long series, so dropping some of the series ordinarily leaves years in which nothing that remains was measured. The result runs from the first year to the last year in which any of the selected series has a measurement. Empty years between those two are kept: an rwl object holds one row per year, and an interior year cannot be dropped without breaking the sequence.

This applies to calls that drop series without indexing years – x[, j], x[j], and subset given only select. Two kinds of call leave the years alone. A call that indexes years returns exactly the years it asks for, measured or not: x[i,], x[i, j], head, tail, common.interval, and subset with its subset argument. A year window is how an rwl object is aligned with something else, such as a climate series or a second collection, and returning fewer years than were asked for would put that alignment out. And a call that keeps every series, such as the reordering x[, order(names(x))] or x[], has dropped nothing to trim for, whatever empty years the object already held.

Two results are not trimmed. A subset holding no measurement in any year is returned whole rather than emptied, and rwl.check reports it. A single series taken with drop = TRUE is a numeric vector, which carries no years at all, so it is returned at full length and stays aligned with the object it came from; use drop = FALSE for a one-series rwl object trimmed to its own years.

subset *drops empty series;* [*keeps them.* Taking years can leave a series with no values at all: in ca533[time(ca533) %in% 1800:1899,], the series CAM132, CAM152, CAM161 and CAM201 end before 1800 and are left as columns of nothing but NA. An empty column is not a series. chron, rwi.stats and the plots skip it, and summary.rwi, interseries.cor, corr.rwl.seg and detrend either drop it or fail on it.

subset and window.rwl therefore drop any series left with no values, and a message names the series dropped. If no series is left with values, that is an error. Use one of these to take a span of years for analysis.

[keeps every series it is asked for, empty or not, on purpose. x[i, j] always returns the columns that j names, in that order, so the result stays lined up with anything indexed alongside it: a vector

of series names, an ids table from read.ids, or a second object with the same series. If [dropped columns, those would fall out of step without a word, and code that indexes with [– dplR's own included – could no longer count on names(x[, j]) being the series it asked for. So row subsets made with [, and head and tail, can return empty series. Drop them with window or subset, or find them with colSums(!is.na(x)) == 0.

Years must stay consecutive. An rwl object holds one row per year in increasing order, and dplR reads the year of a measurement from its row name, so time.rwl, plot.rwl, detrend, chron, rwl.stats and the crossdating functions all take the next row to be the next year. A subset such as x[c(1, 5, 9),] leaves that untrue. Where row subsetting leaves years that are not consecutive and increasing, because rows were taken from the middle, repeated, or reordered, the result is returned as a plain data.frame, with a warning, and without the provenance record. The measurements are unchanged; what is withdrawn is the claim that they are a set of dated series.

The provenance record is carried and cut to fit. The "dplR.provenance" attribute attached by read.tucson follows the subset. What describes the file – its name, the reader, the header lines – is kept. What describes series – the precision of each, the renames, the interior gaps, the problems found while parsing – is kept for the series that remain, and gaps are dropped where they fall outside the years that remain. mixed.precision is recomputed, since a file measured at two precisions can be subset down to series measured at one. A subset element is added, holding the series in the result, its first and last year, and all.series, which is TRUE when every series the file held is still present. A record that cannot be matched to a series in the result is dropped rather than kept, so an object whose series have been renamed since it was read carries a record that says nothing about them in preference to one that says something wrong.

common.interval cuts an object down to the years its series *share*, which is a different question from the years they cover between them.

Value

An object of class c("rwl", "data.frame"), or a numeric vector when one column is selected and drop is TRUE, or a data.frame when row subsetting has left the years no longer consecutive. When series are dropped and years are not indexed, the result runs only over the years the remaining series cover. [returns every series asked for, including any with no values; subset drops those, with a message.

Author(s)

Andy Bunn

See Also

read.rwl, read.tucson, rwl.check, common.interval, time.rwl, subset.data.frame

Examples

```
library(utils)
data(ca533)

## Series are columns and years are rows. Selecting series drops the years
## that none of them covers: ca533 begins in 626, but these five series do
## not.
```

```
ca533.sub <- ca533[, 1:5]
class(ca533.sub)
range(time(ca533))
range(time(ca533.sub))

## Naming years returns those years, whether or not they hold data.
dim(ca533[time(ca533) %in% 1000:1099, 1:5])

## subset() works the same way.
range(time(subset(ca533, select = 1:5)))

## Years are subset by name or by time().
ca533.1800s <- ca533[time(ca533) %in% 1800:1899, ]
range(time(ca533.1800s))

## Four series end before 1800. `[` keeps them as empty columns, so the
## result still has the 34 series asked for; subset() and window() drop
## them and say which.
ncol(ca533.1800s)
names(ca533.1800s)[colSums(!is.na(ca533.1800s)) == 0]
ncol(subset(ca533, time(ca533) %in% 1800:1899))
ncol(window(ca533, 1800, 1899))

## A single series comes back as a vector unless drop is FALSE.
class(ca533[, 1])
class(ca533[, 1, drop = FALSE])

## The provenance record of the data follows the subset.
data(wa082)
gap.series <- attr(wa082, "dplR.provenance")$gaps$series
wa082.sub <- wa082[, gap.series, drop = FALSE]
attr(wa082.sub, "dplR.provenance")$gaps
attr(wa082.sub, "dplR.provenance")$subset$all.series

## Rows taken out of the middle break the one-row-per-year promise that
## dplR relies on. This warns, and gives back a data.frame.
not.rwl <- ca533[c(1, 5, 9), ]
class(not.rwl)
```

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

Description

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

Usage

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

Arguments

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

Details

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

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

spline, approx

Examples

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

```
class(foo) <- c("rwl","data.frame")
fill.internal.NA(foo, fill=0)

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

## note differences in method "Spline" vs. "Linear"
yrs <- time(foo)
plot(yrs, foo$x7, type="b", lwd=3)
lines(yrs, bar$x7, col="red", lwd=2)
lines(yrs, baz$x7, col="green", lwd=1)
```

`gini.coef` *Calculate the Gini Coefficient*

Description

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

Usage

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

Arguments

`x`	a `numeric` vector

Details

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

Value

the Gini coefficient.

Note

The equivalence between the Lorenz-curve formula used in the C code and the relative-mean-difference definition of the Gini coefficient is derived in the vignette: `vignette("math-dplR", package = "dplR")`.

Author(s)

Mikko Korpela, based on original by Andy Bunn

References

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

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

See Also

rwl.stats

Examples

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

---

glk *Calculate Gleichläufigkeit*

Description

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

Usage

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

Arguments

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

Details

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

The probability of exceedence (p) for the Gleichläufigkeit expresses the chance that the Gleichläufigkeit is incorrect. The observed value of the Gleichläufigkeit is converted to a z-score and based

on the standard normal curve the probability of exceedence is calculated. The result is a matrix of all p-values (Jansma 1995, 60-61, see also Visser 2020).

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

Value

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

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

The matrices can be extracted from the list by selecting the name or index number. If two curves have less than 3 years of overlap, Gleichläufigkeit cannot be computed, and `NA` is returned. To calculate the global glk of the dataset `mean(x$glk_mat, na.rm = TRUE)`. See examples for calculating the the global glk without self-similarities (i.e., the ones on the diagonal).

Author(s)

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

References

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

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

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

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

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

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

See Also

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

Examples

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

---

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

---

Description

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

Usage

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

Format

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

Source

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

References

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

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

Description

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

Usage

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

Format

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

Source

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

References

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

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

Description

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

Usage

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

Format

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

Source

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

References

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

---

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

Description

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

Usage

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

Format

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

Source

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

References

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

`hanning` *Hanning Filter*

Description

Applies a Hanning filter of length n to x.

Usage

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

Arguments

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

Details

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

Value

A filtered vector.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

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

See Also

`filter`

Examples

```
library(graphics)
library(utils)
data(ca533)
yrs <- time(ca533)
y <- ca533[, 1]
not.na <- !is.na(y)
yrs <- yrs[not.na]
y <- y[not.na]
plot(yrs, y, xlab = "Years", ylab = "Series1 (mm)",
     type = "l", col = "grey")
```

```
lines(yrs, hanning(y, n = 9), col = "red", lwd = 2)
lines(yrs, hanning(y, n = 21), col = "blue", lwd = 2)
legend("topright", c("Series", "n=9", "n=21"),
       fill=c("grey", "red", "blue"))
```

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

Description

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

Usage

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

Arguments

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

Details

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

A series with no values at all is dropped, with a message naming it, before any series is shown, and its name is dropped from *y.name* too.

Value

An object of class `c("rwi", "data.frame")` (see `as.rwi`) containing each detrended series according to the method used as columns and rownames set to `colnames(`*y*`)`. These are typically years. The method chosen for each series is recorded, named by series, in `attr(x, "dplR.detrend")$method`. Plots are also produced as the user chooses the detrending methods through keyboard input.

Author(s)

Andy Bunn

See Also

detrend.series

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

A vector containing the detrended series (y) according to the method used with names set to colnames(y). These are typically years. The name of the method chosen is attached as attr(x, "method"). A plot is also produced and the user chooses a method through keyboard input.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

detrend.series

`insert.ring` *Edit a Ring-Width Series*

Description

Insert or delete rings from a ring-width series

Usage

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

Arguments

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

Details

Simple editing of ring widths.

Value

A named vector.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

dplR

Examples

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

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

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

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

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

Description

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

Usage

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

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
nyrs	NULL or a number greater than zero. If not NULL, each series is divided by a smoothing spline with this rigidity (see caps) to remove low-frequency variation, as detrend does with method = "Spline". A value of 1 or less is taken as a proportion of each series' length. Unlike the hanning filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
prewhiten	logical flag. If TRUE each series is whitened using ar.
ar.order.max	NULL or a positive integer giving the maximum order of the ar model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If NULL, the order is chosen by AIC up to the ar default, which on long series can be 20 or more. Requires *prewhiten* = TRUE.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
method	Can be either "pearson", "kendall", or "spearman" which indicates the correlation coefficient to be used. Defaults to "spearman". See cor.test.

Details

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

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

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

Value

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

Author(s)

Andy Bunn, patched and improved by Mikko Korpela

See Also

rwl.stats, rwi.stats

Examples

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

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

---

latexDate *Date Conversion to Character in LaTeX Format*

Description

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

Usage

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

Arguments

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

Value

A character vector

Author(s)

Mikko Korpela

Examples

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

---

latexify *Convert Character Strings for Use with LaTeX*

Description

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

Usage

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

Arguments

`x`	a `character` vector
`doublebackslash`	a `logical` flag. If `TRUE`, backslashes in the output are doubled. It seems that Sweave needs `TRUE` and knitr `FALSE`.
`dashdash`	a `logical` flag. If `TRUE` (the default), consecutive dashes ("-") in the input will be rendered as separate characters. If `FALSE`, they will not be given any special treatment, which will usually mean that two dashes are rendered as an en dash and three dashes make an em dash.
`quotes`	a `character` string specifying how single and double quotes (ASCII codes 39 and 34) and stand-alone grave accents (ASCII code 96) in the input are treated. The default is to use straight quotes and the proper symbol for the grave accent. The other option is to use curved right side (closing) quotes, and let LaTeX convert the grave accent to opening curved quotes. Straight double quotes are not available in the default OT1 font encoding of LaTeX. Straight single quotes and the grave accent symbol require the "textcomp" package. See *packages*.

packages — a character vector specifying the LaTeX packages allowed. The use of some symbols in LaTeX requires commands or characters made available by an add-on package. If a package required for a given character is not marked as available, a fallback solution is silently used. For example, curved quotes may be used instead of straight quotes. The supported packages are "eurosym" (not used by default), "fontenc" and "textcomp". Including "fontenc" in the vector means that some other encoding than OT1 is going to be used. Note that straight quotes are an exception in the sense that a reasonable substitute (curved quotes) is available. In many other cases, "textcomp" and "fontenc" are silently assumed.

Details

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

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

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

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

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

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

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

Value

A character vector

Author(s)

Mikko Korpela

References

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

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

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

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

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

Examples

```
x1 <- "clich\xe9\nma\xf1ana"
Encoding(x1) <- "latin1"
x1
x2 <- x1
Encoding(x2) <- "bytes"
x2
x3 <- enc2utf8(x1)
testStrings <-
    c("different     kinds\nof\tspace",
      "control\a characters \ftoo",
      "{braces} and \\backslash",
      '#various$ %other^ &characters_ ~escaped"/coded',
      x1,
      x2,
      x3)
latexStrings <- latexify(testStrings, doublebackslash = FALSE)
## All should be "unknown"
Encoding(latexStrings)
cat(latexStrings, sep="\n")
## Input encoding does not matter
identical(latexStrings[5], latexStrings[7])
```

`morlet` *Perform a Continuous Morlet Wavelet Transform*

Description

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

Usage

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

Arguments

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

Details

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

Value

A list containing:

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

Note

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

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

See Also

wavelet.plot

Examples

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

---

net *Calculate NET*

Description

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

Usage

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

Arguments

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

Details

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

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

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

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

Value

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

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

Author(s)

Mikko Korpela

References

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

Examples

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

## End(Not run)
```

---

nm046 *Los Alamos Tree Ring Widths*

Description

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

Usage

```
data(nm046)
```

Format

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

Source

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

References

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

Examples

```
library(utils)
data(nm046)

## Where the data came from. read.tucson() records what it saw and attaches
## it to what it returns; this data set carries that record.
prov <- attr(nm046, "dplR.provenance")
prov$file          # the archived file it was read from
prov$precision     # the precision each series was measured at
prov$gaps          # interior gaps, if any -- this file has none
prov$events        # problems found while reading -- none here either

## rwl.report() puts the file and precision at the head of its report
rwl.report(nm046)
```

---

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

Description

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

Usage

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

Arguments

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

Details

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

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

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

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

Value

A filtered vector.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

hanning, detrend

Examples

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

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

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

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

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

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

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

Description

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

Usage

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

Arguments

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

Details

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

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

chron

Examples

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

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

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

---

plot.crs *Plotting crs Objects*

Description

Plots objects returned from corr.rwl.seg.

Usage

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

Arguments

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

Details

Each segment is drawn in blue if it correlates above the critical value, red if it does not, and green where it does not overlap the bin completely. If corr.rwl.seg was run with *lag.max* greater than 0, segments that correlate better at another lag (COFECHA's B flag) are drawn in purple instead, whatever their correlation as dated.

Value

None. A plot is produced.

Author(s)

Andy Bunn

See Also

corr.rwl.seg

Examples

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

---

plot.rwl *Plotting Rwl Objects*

Description

Plots rwl objects.

Usage

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

Arguments

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

Value

None. A plot is produced.

Author(s)

Andy Bunn

See Also

read.rwl

Examples

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

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Mikko Korpela

See Also

wc.to.po, rcs, write.tridas

Examples

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

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

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

Details

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

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

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

Value

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

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

Author(s)

Pierre Mérian. Improved by Mikko Korpela and Andy Bunn.

References

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

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

See Also

`skel.plot`

Examples

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

---

powt *Power Transformation of Tree-Ring Data*

Description

Power transformation of tree-ring width.

Usage

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

Arguments

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

Details

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

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

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

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

Value

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

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

Author(s)

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

References

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

Examples

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

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

---

print.redfit *Printing Redfit Results*

Description

Print information contained in or derived from a redfit object.

Usage

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

Arguments

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

Value

Invisibly returns *x*.

Author(s)

Mikko Korpela

References

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

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

See Also

redfit

Examples

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

---

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

Description

This function prints the results of `rwl.report`

Usage

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

Arguments

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

Details

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

Value

Invisible

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

rwl.report, summary.rwl, interseries.cor

Examples

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

---

rasterPlot *Add Raster Elements to Plot*

Description

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

Usage

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

Arguments

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

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

Details

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

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

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

Value

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

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

Note

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

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

Author(s)

Mikko Korpela

Examples

```
library(graphics)
library(stats)

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

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

```
par(parnew)

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

---

rcs *Regional Curve Standardization*

Description

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

Usage

```
rcs(rwl, po = NULL, nyrs = NULL, f = 0.5, biweight = TRUE, ratios = TRUE,
    rc.out = FALSE, make.plot = TRUE, method = c("caps", "ads"),
    min.n = NULL, pos.slope = TRUE, ...)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl
po	a data.frame containing two variables. Variable one (*series* in the example below) gives the series ID as either characters or factors. These must exactly match colnames(*rwl*). Variable two (*pith.offset* in the example below) must be integral values and give the years from the beginning of the core to the pith (or center) of the tree. The minimum value is 1. If NULL (the default) a pith offset of 1 is assumed for all series, meaning each series is treated as beginning at cambial age 1. This is almost certainly unrealistic for most datasets but is provided as a convenience when pith offset data are unavailable.
nyrs	a number giving the rigidity of the smoothing spline. For method = "caps" this defaults to 0.1 of the length of the regional curve if NULL. For method = "ads" this sets the starting stiffness (nyrs0) and defaults to 50 if NULL.

f	a number between 0 and 1 giving the frequency response or wavelength cutoff. Defaults to 0.5.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
ratios	logical flag. If TRUE (the default) then indices are calculated by division, if FALSE indices are calculated by subtraction.
rc.out	logical flag. Returns the regional curve along with the ring-width indices if TRUE.
make.plot	logical flag. Makes a plot of the raw ring-width series by cambial age, the biweight mean, and the regional curve if TRUE.
method	a character string giving the smoothing method to apply to the mean regional curve. Either "caps" (the default, a cubic smoothing spline via caps) or "ads" (an age-dependent spline via ads).
min.n	an optional integer giving the minimum sample depth required at a given cambial age for that age to be included when fitting the regional curve. Ages where fewer than *min.n* series contribute are excluded from the tail of the curve, and indices for rings falling beyond the truncated curve are returned as NA. If NULL (the default) the full curve is used.
pos.slope	a logical flag passed to ads. If TRUE (the default) a positive slope is allowed at the end of the fitted curve. Ignored when *method* = "caps".
...	other arguments passed to plot.

Details

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

Two smoothers are available for fitting the regional curve. The default (method = "caps") uses a cubic smoothing spline where the frequency response is 0.50 at a wavelength of 10 percent of the maximum cambial age unless specified differently via *nyrs* and *f* (see caps). The alternative (method = "ads") uses an age-dependent spline whose stiffness increases with cambial age, which may better preserve low-frequency variability in the biological trend (see ads).

The *min.n* argument can be used to truncate the regional curve where sample depth becomes thin, preventing poorly replicated tail ages from influencing the fit.

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.

A series with no values at all is dropped, with a message naming it, and the rest are standardized as if it had never been there.

Value

An object of class c("rwi", "data.frame") (see as.rwi) containing the dimensionless and detrended ring-width indices with column names, row names and dimensions of *rwl*. If *rc.out* is TRUE

then a `list` will be returned with a `data.frame` containing the detrended ring widths as above and a `vector` containing the regional curve.

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. Extended with `method`, `min.n`, and `pos.slope` arguments by Andy Bunn.

References

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

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

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

See Also

`detrend`, `chron`, `cms`, `caps`, `ads`

Examples

```
library(utils)
data(gp.rwl)
data(gp.po)

## Basic use: caps smoother (default), return indices only
gp.rwi <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
              make.plot = FALSE)
str(gp.rwi)

## Return the regional curve alongside the indices
gp.out <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
              rc.out = TRUE, make.plot = FALSE)
str(gp.out)

## Plot the regional curve with a title
rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
    rc.out = FALSE, make.plot = TRUE, main = "Regional Curve")

## Use the age-dependent spline smoother instead of caps
gp.rwi.ads <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
                  method = "ads", rc.out = FALSE, make.plot = FALSE)

## Truncate the regional curve where fewer than 5 series contribute.
## Indices for rings beyond the truncated curve are returned as NA.
gp.out.trunc <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
                    method = "ads", min.n = 5,
                    rc.out = TRUE, make.plot = FALSE)
```

```
## Compare the two smoothers visually
op <- par(no.readonly = TRUE)
par(mfrow = c(1, 2))
rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
    method = "caps", make.plot = TRUE, main = "caps")
rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
    method = "ads", make.plot = TRUE, main = "ads")
par(op)

## Compare caps and ads indices for one series
gp.caps <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
               method = "caps", rc.out = FALSE, make.plot = FALSE)
gp.ads  <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
               method = "ads",  rc.out = FALSE, make.plot = FALSE)
cor(gp.caps[[1]], gp.ads[[1]], use = "complete.obs")

## Use subtraction rather than division (e.g. after variance stabilization)
gp.rwi.sub <- rcs(rwl = gp.rwl, po = gp.po, biweight = TRUE,
                  ratios = FALSE, make.plot = FALSE)
```

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Mikko Korpela

See Also

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

Examples

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

## End(Not run)
```

---

read.crn *Read Tucson Format Chronology File*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

---

read.fh *Read Heidelberg Format Ring Width File*

Description

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

Usage

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

Arguments

`fname`	a `character` vector giving the file name of the fh file.
`BC_correction`	a `logical`. If TRUE the correction moves BC-dates one year forward.
`encoding`	the encoding of the file, used only if the file is not valid UTF-8. Any name `iconv` accepts will do, e.g. `"latin1"` or `"ISO-8859-2"`. `NULL`, the default, means no encoding has been declared: the file is then read as `"latin1"` and a warning says so, naming the line and quoting the text the assumption produced. This matters more for this format than for most: Heidelberg files are widely used in European dendroarchaeology and their `HEADER` blocks carry free text – site, species, project and operator names – which is often not ASCII and often not UTF-8. The tiers, and why the fallback does not run a charset detector, are described under 'Encoding' in `read.tucson`.

Details

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

Value

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

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

Author(s)

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

References

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

See Also

`read.rwl`

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

Description

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

Usage

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

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

Arguments

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

use.cor — a logical flag. If TRUE and *stc* is "auto", correlation clustering may be used for determining the length of the tree and core parts. See Details.

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

ignore.site.case — a logical flag. If TRUE, the function does not distinguish between upper case and lower case letters in the site part of the series names.

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

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

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

Details

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

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

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

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

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

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

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

- Numbers in tree part, core part starts with something else
- Alphabets in tree part, core part starts with something else
- Alphabets, either tree part all lower case and core part all upper case or vice versa. For this to work, *ignore.case* must be set to `"auto"` or `FALSE`.
- All digits. In this case, the number of characters belonging to the tree and core parts is detected with one of the following methods.
  - If numeric tree parts were found before, it is assumed that the core part is missing (one core per tree).
  - It the series are numbered continuously, one core per tree is assumed.
  - Otherwise, try to find a core part as the suffix so that the cores are numbered continuously.

  If none of the above fits, the tree / core split of the all-digit names will be decided with the methods described further down the list, or finally with the fallback mechanism.
- The combined tree / core part is empty or one character. In this case, the core part is assumed to be missing.
- Tree and core parts separated by a punctuation or white space character

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

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

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

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

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

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

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

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

Value

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

Author(s)

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

See Also

`rwi.stats`, `read.rwl`

Examples

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

`read.rwl` *Read Ring Width File*

Description

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

Usage

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

Arguments

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

Details

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

With `format = "auto"` the file's first lines are scanned and the most specific match wins: a DPL compact column specification or a `HEADER:` line, then a `<tridas>` element, then a NOAA/NCEI template header, then a comma separated sheet, and Tucson if nothing else matches. A sheet is recognised by every line splitting into the same number of comma separated fields, not by the presence of a comma. Only commas are looked for: a tab separated file may be a NOAA template, so tab separated sheets need `format = "sheet"` with a `sep` argument.

`format = "sheet"` is handled by `read.sheet`, which as of `dplR` 1.8.0 replaces the deprecated `csv2rwl`. Files that `csv2rwl` accepted may now be refused; see `read.sheet`.

`"csv"` is accepted as a synonym for `"sheet"` because it has been a valid value of `format` since before `read.sheet` existed. It is the less accurate of the two: `read.sheet` also reads tab, semicolon and pipe separated files, so `format = "sheet"` with a `sep` argument is an ordinary call. Prefer `"sheet"` in new code.

Value

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

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

Author(s)

Mikko Korpela

See Also

read.tucson, read.tucson.legacy, read.compact, read.tridas, read.fh, read.sheet, write.rwl

---

read.sheet *Read Ring Widths from a Spreadsheet File*

Description

Read a file of ring widths saved in "spreadsheet" layout: years down the rows, series across the columns, and the years in the first column.

Usage

```
read.sheet(fname, sep = NULL, dec = ".", layout = c("wide", "long"),
           transpose = FALSE, comment.char = "#", fill.internal.NA = NULL,
           fix.dup.char = "X", encoding = NULL, verbose = TRUE,
           strict = FALSE, ...)

csv2rwl(fname, ...)
```

Arguments

fname	a character vector giving the file name.
sep	the field separator: one of ",", "\t", ";" or "|", or any single character. NULL, the default, detects it.
dec	the decimal mark, "." or ",". A decimal comma is detected automatically when the separator is a semicolon.
layout	the shape of the file. "wide", the default, is years down the rows and series across the columns. "long" is a "long" or "tidy" file of series, year, value triples. Note that this is unrelated to the long argument of read.crn and read.tucson, which refers to a wider fixed-width year field.
transpose	logical flag. Set to TRUE for a file with the series in rows and the years in columns.
comment.char	lines beginning with this string are treated as a comment header: skipped, and kept in the provenance record.
fill.internal.NA	passed to fill.internal.NA to fill interior gaps. The default, NULL, leaves them as NA.
fix.dup.char	character string appended to a series ID that appears more than once in the header row.

encoding	the encoding of the file, used only if the file is not valid UTF-8. Any name iconv accepts will do, e.g. "latin1" or "ISO-8859-2". NULL, the default, means no encoding has been declared: the reader then falls back to "latin1" and says so. The tiers, and why the fallback does not run a charset detector, are described under 'Encoding' in read.tucson; read.sheet behaves identically.
verbose	logical flag. Write a summary of the file to the console.
strict	logical flag. If TRUE, every recoverable problem becomes an error instead of a warning.
...	other arguments passed to fread.

Details

The expected layout is the one a spreadsheet produces: the first column holds the years, the first row holds the series IDs, and each remaining cell holds one ring width.

Year	Ser1A	Ser1B	Ser2A	Ser2B
1901	NA	0.45	0.43	0.24
1902	NA	0.05	0.00	0.07
1903	0.17	0.46	0.03	0.21
1904	0.28	0.21	0.54	0.41
etc...				

An empty cell, NA, NaN, "." and "-" are all read as missing. Any other value that does not parse as a number is an error rather than a missing value, and the offending cells are named.

Unlike a Tucson file, a spreadsheet has no format to conform to, so this function checks the contents rather than assuming them. The years must be whole numbers, ascending, unique and consecutive: a year with no measurements is a row of NA, not a missing row. Every column after the first must be numeric. Series IDs are read verbatim, and anything that changes one – a stripped byte order mark, a resolved duplicate – is reported and recorded rather than done silently.

Separators:

Comma, tab, semicolon and pipe separated files are all read. Left at NULL, sep is detected by looking for the candidate that gives every line the same number of fields, preferring the one that gives the most; quoted spans are ignored while counting, so a series ID may contain the separator. The detected separator is recorded in the provenance record as SEP_GUESS and printed when verbose is TRUE, but it is not a warning: detection is the default, and a defect it is not.

A semicolon separated file is usually a European export and carries decimal commas. When the separator is detected as a semicolon and a digit-comma-digit appears in the data, dec is set to "," and recorded as DEC_COMMA. Thousands separators are not handled: "1.234,5" and "1,234.5" are the same characters under two conventions and nothing in the file says which.

sep and dec may not be the same character.

Long format:

With layout = "long" the file holds one row per observation rather than a rectangle: a series ID, a year and a value. Columns are taken by position, which is what write.sheet writes; if the header names all three of series, year and value, in any case and any order, the names are used instead.

The year span of the result runs from the earliest to the latest year in the file. A year with no observation in any series therefore comes back as a row of NA, and is reported, since years with no data break several dplR functions. One series may hold only one value per year.

`read.sheet` refuses a NOAA/NCEI template file. Those files are a commented metadata block followed by a tab separated data table, and reading the table alone would discard the coordinates, species, investigators and DOI above it.

Value

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

The object carries an attribute `"dplR.provenance"`, a `list` in the same shape as the one attached by `read.tucson`, recording what the reader found and what it changed. Its `events` element is a `data.frame` of findings, each with a stable id such as `"ID_RENAMED"` or `"UNITS_SUSPECT"`.

Deprecated

`csv2rwl` is deprecated as of `dplR` 1.8.0 and now calls `read.sheet`. It will be made defunct in a later release.

This is a change of behaviour and not only a change of name. `csv2rwl` assigned the class `"rwl"` directly, which bypassed `as.rwl` and its requirement that the row names be consecutive integers. It therefore accepted files that `read.sheet` refuses:

- non-consecutive or duplicated years, which produced an object whose `time` was not a continuous span;
- columns holding text, which travelled inside the `rwl` until some later function failed on them;
- series IDs were passed through `read.table(check.names = TRUE)`, so `"1A"` became `"X1A"` and `"LF-2B"` became `"LF.2B"`, silently.

A file that `csv2rwl` read may therefore now stop with an error. That is the deprecation doing its job: the object it used to return was invalid. `csv2rwl` also stopped when a series ID appeared twice, where `read.sheet` renames the duplicate and records the change, as `read.tucson` does.

There is no `csv2rwl.legacy`. `read.tucson.legacy` exists because the behaviour of the old Tucson reader differs from the new one in ways that are occasionally wanted; that is not the case here.

Author(s)

Andy Bunn

See Also

`write.sheet`, `read.rwl`, `read.tucson`, `rwl.check`, `as.rwl`

Examples

```
library(utils)
data(ca533)
# write out a sheet that read.sheet will understand
tm <- time(ca533)
foo <- data.frame(tm, ca533, check.names = FALSE)
names(foo)[1] <- "Year"
tmpName <- tempfile(fileext = ".csv")
write.csv(foo, file = tmpName, row.names = FALSE)

# read it back in
bar <- read.sheet(tmpName, verbose = FALSE)
# the provenance record travels with the data
str(attr(bar, "dplR.provenance")$events)

unlink(tmpName)
```

---

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

Description

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

Usage

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

Arguments

fname	character vector giving the file name of the TRiDaS file.
ids.from.titles	logical flag indicating whether to override the (tree, core, radius, measurement) structure imposed by the element hierarchy (element, sample, radius, measurementSeries) of the file. If TRUE, measurement series will be rearranged by matching titles in the file at the aforementioned four levels of the hierarchy. Defaults to FALSE, i.e. the element hierarchy of the file will be used.
ids.from.identifiers	logical flag indicating whether to (partially) override the element hierarchy of the file. If TRUE, measurement series will be grouped according to matching identifiers at the measurementSeries level, where identifiers are available. The changes caused by this option are applied on top of the structure imposed by the file or computed from matching titles if *ids.from.titles* == TRUE. Defaults to TRUE.

`combine.series` `logical` flag indicating whether to combine two or more measurement series with the same set of (tree, core, radius, measurement) ID numbers. Each set of combined measurement series will be represented by one column of a resulting `data.frame`. Overlapping years of combined series do not produce a warning. If several data points are available for a given year, the function chooses one in a rather arbitrary manner. This option can only have effect when *ids.from.titles* `||` *ids.from.identifiers*.

`trim.whitespace` `logical` flag indicating whether to replace repeated white spaces in the text content of the file with only one space. Defaults to `TRUE`, i.e. excess white space will be trimmed from the text.

`warn.units` `logical` flag indicating whether to warn about unitless measurements and "strange" units. The function expects measurements in units that can be converted to millimetres. Defaults to `TRUE`: warnings will be given. For example, density measurements will trigger warnings, which can be disabled by setting this option to `FALSE`.

Details

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

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

Value

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

`measurements` A `data.frame` or a list of `data.frames` with the series in columns and the years as rows. Contains measurements ('`<measurementSeries>`') with known years. The series IDs are the column names and the years are the row names. The series IDs are derived from '`<title>`' elements in the input file. Each unique combination of '`<project>`', '`<object>`', '`<unit>`', '`<taxon>`', and '`<variable>`' gets a separate `data.frame`.

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

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

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

`wood.completeness` A `data.frame` or a list of `data.frames` containing wood completeness information. Column names are a subset of the following, almost self-explanatory

set: "pith.presence", "heartwood.presence", "sapwood.presence", "last.ring.presence", "last.ring.details", "bark.presence", "n.sapwood", "n.missing.heartwood", "n.missing.sapwood", "missing.heartwood.foundation", "missing.sapwood.foundation", "n.unmeasured.inner", "n.unmeasured.outer".

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

text The text of the '`type`' element

lang The language of the text

normal The normalized name of the type

normalId The ID value of the type in the standard dictionary

normalStd The name of the standard

project.id The ID of the project

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

tree.id The ID of the tree

core.id The ID of the core

derived.id The ID of the derived series

project.title The title of the project

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

tree.title The title of the tree

core.title The title of the core

derived.title The title of the derived series

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

text The text of the '`comments`' element

project.id The ID of the project

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

tree.id The ID of the tree

core.id The ID of the core

radius.id The ID of the radius

measurement.id The ID of the measurement series

derived.id The ID of the derived series

project.title The title of the project

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

tree.title The title of the tree

core.title The title of the core

radius.title The title of the radius

measurement.title The title of the measurement series

derived.title The title of the derived series

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

text The text of the '`identifier`' element

domain The domain which the identifier is applicable to

project.id The ID of the project

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

tree.id The ID of the tree

core.id The ID of the core

radius.id The ID of the radius

measurement.id The ID of the measurement series

derived.id The ID of the derived series

project.title The title of the project

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

tree.title The title of the tree

core.title The title of the core

radius.title The title of the radius

measurement.title The title of the measurement series

derived.title The title of the derived series

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

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

text The remark

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

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

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

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

text The remark

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

idx Index to a value in the series

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

text The remark

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

idx Index to a value in the series

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

name Name of the laboratory

acronym Acronym of the name

identifier Identifier

domain Domain which the identifier is applicable to

addressLine1 Address

addressLine2 Another address line

cityOrTown City or town

stateProvinceRegion State, province or region

postalCode Postal code

country Country

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

identifier Identifier

domain Domain which the identifier is applicable to

description General description

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

metres The altitude in metres

project.id The ID of the project

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

tree.id The ID of the tree

project.title The title of the project

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

tree.title The title of the tree

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

idRef Reference to a series in the same file

xLink URI

identifier Identifier of a series not necessarily in the same file

domain Domain which the identifier is applicable to

project.id The ID of the project

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

tree.id The ID of the tree

project.title The title of the project

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

tree.title The title of the tree

Note

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

Author(s)

Mikko Korpela

References

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

See Also

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

read.tucson *Read Tucson Format Ring Width File*

Description

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

Usage

```
read.tucson(fname, header = NULL, long = FALSE,
            encoding = getOption("encoding"),
            edge.zeros = TRUE, verbose = TRUE,
            comment.char = "#", fix.dup.char = "X",
            fill.internal.NA = NULL, strict = FALSE)
```

Arguments

fname	a character vector giving the file name of the rwl file.
header	ignored, and accepted only so that existing calls do not fail. Header lines are now identified per line by their content. See 'Details'.
long	ignored, and accepted only so that existing calls do not fail. The two possible column layouts are now detected per line, so 8-character series IDs and years before -999 may occur in the same file. As in read.crn, long here means a wider fixed-width year field; it is unrelated to the "long" or "tidy" layout of read.sheet, which is selected with layout = "long". See 'Details'.
encoding	the encoding of the file, used only if the file is not valid UTF-8. Any name iconv accepts will do, e.g. "latin1" or "ISO-8859-2". The default, getOption("encoding"), means no encoding has been declared: the reader then falls back to "latin1" and says so. See 'Encoding' below.
edge.zeros	logical flag indicating whether leading or trailing zeros in series will be preserved (when the flag is TRUE, the default) or discarded, i.e. marked as NA (when FALSE).
verbose	logical flag, print info on data. When TRUE (the default) every interior gap is listed with the series, the years, and what the file actually held there.
comment.char	a character giving the character that marks a comment line. Lines containing it are dropped.
fix.dup.char	a character appended to a series ID when the file contains more than one record under that ID and the records overlap in time. Appended repeatedly if necessary, so that no measurement is lost and no name collides.
fill.internal.NA	what to do with interior gaps, i.e. years inside a series for which the file records no measurement. The default, NULL, leaves them as NA. Any other value is passed to fill.internal.NA, so 0 reproduces the output of read.tucson.legacy, and "Mean", "Spline" or "Linear" interpolate. See 'Details'.

`strict` `logical` flag. When `FALSE`, the default, every recoverable problem is reported as a warning and reading continues. When `TRUE` the same problems become errors, so that a pipeline can refuse a file rather than carry a warned guess forward.

Details

This reads in a standard rwl file as defined according to the standards of the ITRDB at https://www.ncei.noaa.gov/pub/data/paleo/treering/treeinfo.txt.

This function was replaced in dplR 1.8.0. The previous implementation is still available, unchanged, as `read.tucson.legacy`. The two differ in at least three important ways.

Interior gaps are no longer filled with zero. This is the change most likely to alter your numbers. Where a file records no measurement for a stretch of years inside a series, the old reader silently wrote zeros. A ring width of zero means a locally absent ring, which is a real observation about a tree in a year; a gap means the ring was not measured, which is a statement about the data. These are different claims and the reader no longer substitutes one for the other. Set `fill.internal.NA = 0` to recover the old values exactly.

Note that several dplR functions, `detrend` among them, do not accept `NA` inside a series and will fail on such data. If you need to detrend a series containing interior gaps you must decide what those gaps mean and fill them deliberately, with `fill.internal.NA` or with `fill.internal.NA = 0` here.

Malformed files are reported rather than guessed at. Where the ten fixed-width columns and a whitespace split of the same line disagree, the line does not conform to the format and nothing in it says which reading was meant, so it is returned as `NA` with both readings reported. Repeated series IDs are reported instead of being silently welded into one series, and are renamed when their records overlap. Tab characters, which have no defined width in a fixed-width format, are expanded to 8-column tab stops and reported.

The header and column layout are determined per line. This is why `header` and `long` are no longer used. `header` could only skip a fixed three lines and so could not describe the 4- and 6-line headers that real files contain. `long` is worse: years before -999 need five columns and so take column 8 away from the series ID, but that varies line by line, so a file mixing 8-character IDs with BC dates is read wrongly whichever way a per-file switch is set. Both are now decided from the content of each line. The arguments are still accepted, in their original positions, so that existing code and `read.rwl`'s pass-through of `...` do not fail with an "unused argument" error; passing either one warns.

Editorial: A closing word on the format itself. The Tucson decadal format dates from the era of the punch card. It is nominally fixed-width, yet files in the ITRDB arrive with tab characters sitting in the measurement fields, and a tab has no defined width at all. Column 8 belongs to the series ID unless the year is earlier than -999, in which case the year claims it, and nothing but a minus sign distinguishes the two readings. A negative number marks missing data, except when it marks the end of a record; `999` ends a record in a 0.01 mm file, so a genuine 9.99 mm ring cannot be told apart from a terminator. A single ID may carry several records written in any order at all: in ITRDB '`ut542.rwl`' a series runs to 1841 and then starts over at 1353 further down the file. A decade line may resume off its boundary and quietly swallow the years it skipped, as '`chil012.rwl`' does at 1242 for series `AGU034`. None of this is the fault of the people who made these data available, and the tree-ring community deserves credit for sharing its measurements openly long before most of science thought to do so. But we are still paying for a format built around eighty columns of cardboard. `read.tucson` tries to read what the file says, to report plainly what it did, and to refuse

rather than guess when a line admits two readings. Even so, and despite our best efforts, this format can and does fail.

Value

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

The returned object carries a `"dplR.provenance"` attribute recording what the reader saw: the header lines, the precision each series was measured at and whether the file mixes them, any series the reader renamed (as `old` and `new`), the interior gaps and what the file held at each, and one row per recoverable problem found while parsing. Most of this cannot be recovered from the returned data or by reading the file again, and `rwl.check` reports on it. Subsetting an `rwl` object with `[.rwl` carries the record along and cuts it down to the series and years that are left, but functions that return a new object, such as `detrend`, drop it, so it describes the read rather than travelling with the data indefinitely.

Encoding

Ring widths are digits, but the header lines and occasionally the series IDs are not, and archived files carry accented site names and investigator names in whatever 8-bit encoding the contributor's machine used. `read.tucson` resolves this in three steps.

First, the file is tested against UTF-8. This is a decision and not a guess: UTF-8 is self-validating, and plain ASCII passes, so virtually every file takes this path and nothing is reported.

If the file is not valid UTF-8 and `encoding` was supplied, that encoding is used and an `ENCODING_DECLARED` event is recorded. If the declared encoding does not fit the file, `read.tucson` stops rather than transcode into replacement characters.

If the file is not valid UTF-8 and no encoding was supplied, it is read as `"latin1"` and an `ENCODING_ASSUMED` event is recorded, with a warning naming the line and quoting the text the assumption produced, so the guess can be judged at a glance. Under `strict = TRUE` this becomes an error.

The fallback guesses rather than running a charset detector because detectors do not work on this kind of file. They are byte-frequency language models and need a volume of non-ASCII text that a file of ring widths never has; on a rwl that is ASCII apart from one accented name, ICU's detector scores the right answer at 0.16 and ranks UTF-16 alongside it. `"latin1"` is used instead because it is total (every byte decodes, so it cannot fail) and lossless (the round trip is byte-identical, so a wrong guess discards nothing and the file can simply be re-read with `encoding` set).

Author(s)

Original reader by Hung Nguyen. Reworked and extended by Andy Bunn.

See Also

`read.tucson.legacy`, `read.rwl`, `read.compact`, `read.tridas`, `read.fh`, `write.tucson`, `fill.internal.NA`, `[.rwl`

Examples

```
## A short file with an interior gap: the 1910s are marked -999,
## i.e. not measured.
tf <- tempfile()
writeLines(c("TST01A  1900   123   134   145   156   167   178   189   150   141   132",
             "TST01A  1910  -999  -999  -999  -999  -999  -999  -999  -999  -999  -999",
             "TST01A  1920   211   222   233   244   255   266   277   288   299   300",
             "TST01A  1930   311   999"), tf)

## The default reports the gap and returns it as NA.
x <- read.tucson(tf)
x["1915", ]

## Ask for the old behaviour explicitly.
x0 <- read.tucson(tf, fill.internal.NA = 0)
x0["1915", ]

unlink(tf)
```

`read.tucson.legacy` *Read Tucson Format Ring Width File (Legacy Reader)*

Description

This function reads in a Tucson (decadal) format file of ring widths (.rwl). It is the reader that dplR used through version 1.7.9, kept unchanged for backward compatibility. New code should use `read.tucson`.

Usage

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

Arguments

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

edge.zeros	logical flag indicating whether leading or trailing zeros in series will be preserved (when the flag is TRUE, the default) or discarded, i.e. marked as NA (when FALSE).
verbose	logical flag, print info on data.

Details

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

This function was the default Tucson reader in dplR through version 1.7.9. As of 1.8.0 that role belongs to read.tucson, which reads files this one cannot, reports malformed lines rather than guessing at them, and by default returns interior gaps as NA instead of filling them with zero. This function is retained so that existing workflows continue to produce exactly the numbers they always have, and because it honours the encoding argument, which read.tucson does not. Use read.tucson(fname, fill.internal.NA = 0) to get the new reader's parsing with the old reader's treatment of gaps.

Value

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

Author(s)

Andy Bunn. Patched and greatly improved by Mikko Korpela.

See Also

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

---

redfit *Red-Noise Spectra of Time-Series*

Description

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

Usage

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

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

Arguments

x	a numeric vector representing a possibly unevenly spaced time-series.
t	a numeric vector of times or ages corresponding to *x*. If missing, this is set to an evenly spaced increasing integer sequence (1, 2, ...) along *x*. See *txOrdered*.
tType	a character string indicating the type of the *t* vector: either times or ages.
nsim	a numeric value giving the number of simulated AR1 spectra to compute.
mctest	a logical flag. If TRUE, performs a Monte Carlo test for computing red noise false-alarm levels. In that case, the result list will contain non-NULL elements "ci80", "ci90", "ci95" and "ci99".
ofac	oversampling factor for Lomb-Scargle Fourier transform. A numeric value.
hifac	maximum frequency to analyze relative to the Nyquist frequency. A numeric value.
n50	number of segments. The segments overlap by about 50 percent.
rhopre	a numeric value giving the prescribed autocorrelation coefficient. If NULL or negative, the autocorrelation coefficient will be estimated from the data.
p	a numeric or bigq (if package "gmp" is installed) vector of significance levels for a statistical test considering the number of runs in a sequence. See 'Details'.
iwin	the type of window used for scaling the values of each segment of data. A numeric value or one of "rectangular", "welch i", "hanning", "triangular" and "blackman-harris". Integers 0:4 correspond to the character values, in this order. The default *iwin* = 2 corresponds to the "hanning" window.
txOrdered	a logical flag. If TRUE, it is assumed that *t* is in ascending order without duplicates. If FALSE (the default), *t* will be sorted and *x* reordered accordingly. Values of *x* at identical values of *t* are averaged. If duplicates are found, the non-duplicated *t* and averaged *x* will be included in the return value of the function.
verbose	a logical flag. If TRUE, the function prints some information while operating.
seed	a value to be given to set.seed at the start of the function. The value is also recorded in the list returned. If not NULL, this can be used for reproducible results.
maxTime	a numeric value giving the approximate maximum time in seconds to use for computing the exact acceptance regions of the number of runs test. See also *nLimit*.
nLimit	a numeric value giving the maximum *n* for which runcrit will try to compute an exact solution to the acceptance regions of the number of runs test. Precomputed solutions may exist for larger *n*). This limit is in place because a part of the exact solution always needs to be computed in order to roughly estimate the total time and whether it would exceed *maxTime*. If *nLimit* is very large, *maxTime* may be (greatly) exceeded while computing the aforementioned part of the exact solution.
n	an integral value giving the length of the sequence in the number of runs test.

Details

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

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

- More precision is used in some constants and computations.
- All the data are used: the last segment always contains the last pair of (t, x). There may be small differences between `redfit` and REDFIT with respect to the number of points per segment and the overlap of consecutive segments.
- The critical values of the runs test (see the description of `runcrit` below) differ between `redfit` and REDFIT. The approximate equations in REDFIT produce values quite far off from the exact values when the number of frequencies is large.
- The user can select the significance levels of the runs test.
- Most of the window functions have been adjusted.
- 6 dB bandwidths have been computed for discrete-time windows.

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

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

Value

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

`varx`	variance of x estimated from its spectrum. A `numeric` value.
`rho`	average autocorrelation coefficient, either estimated from the data or prescribed (*rhopre*). A `numeric` value.
`tau`	the time scale of an AR1 process corresponding to *rho*. A `numeric` value.
`rcnt`	a `numeric` value giving the number of runs to be used for a statistical test studying the difference between a theoretical AR1 spectrum and the bias-corrected spectrum estimated from the data. Null hypothesis: the two spectra agree, i.e. the probability of either being larger than the other is 0.5 at every point. Requires that *iwin* `== 0` (`"rectangular"`), *ofac* `== 1` and *n50* `== 1`. Otherwise the test is not performed and *rcnt* is `NULL`.

`rcritlo` a `numeric` vector of critical low values for the number of runs, i.e. the lowest value for accepting the null hypothesis at each level of significance p. When returned from `redfit`, `NULL` when *rcnt* is `NULL`.

`rcrithi` a `numeric` vector of critical high values for the number of runs, i.e. the highest value for accepting the null hypothesis at each level of significance p. When returned from `redfit`, `NULL` when *rcnt* is `NULL`.

`rcritexact` a `logical` vector specifying whether each pair of *rcritlo* and *rcrithi* are exact values (`TRUE`) or approximated from a normal distribution (`FALSE`). When returned from `redfit`, `NULL` when *rcnt* is `NULL`.

`freq` the frequencies used. A `numeric` vector. The other numeric vectors have the same length, i.e. one value for each frequency studied.

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

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

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

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

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

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

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

`ci95` 95th percentile red noise spectrum.

`ci99` 99th percentile red noise spectrum.

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

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

np number of points in the data.

nseg number of points in each segment.

nfreq number of frequencies in the results.

avgdt average sampling interval.

df difference between consecutive frequencies.

fnyq average Nyquist frequency.

n50 value of the *n50* argument.

ofac value of the *ofac* argument.

hifac value of the *hifac* argument.

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

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

nsim value of the *nsim* argument.

mctest value of the *mctest* argument.

rhopre value of the *rhopre* argument.

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

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

Author(s)

Mikko Korpela. Examples by Andy Bunn.

References

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

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

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

See Also

print.redfit

Examples

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

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

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

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

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

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

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

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

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

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

plot(redf.x[["freq"]], redf.x[["gxxc"]],
     ylim = range(redf.x[["ci99"]], redf.x[["gxxc"]]),
     type = "n", ylab = "Spectrum", xlab = "Frequency (1/yr)",
     axes = FALSE)
grid()
lines(redf.x[["freq"]], redf.x[["gxxc"]], col = "#1B9E77")
lines(redf.x[["freq"]], redf.x[["ci99"]], col = "#D95F02")
lines(redf.x[["freq"]], redf.x[["ci95"]], col = "#7570B3")
lines(redf.x[["freq"]], redf.x[["ci90"]], col = "#E7298A")
freqs <- pretty(redf.x[["freq"]])
pers <- round(1 / freqs, 2)
```

```
axis(1, at = freqs, labels = TRUE)
axis(3, at = freqs, labels = pers)
mtext(text = "Period (yr)", side = 3, line = 1.1)
axis(2); axis(4)
legend("topright", c("x", "CI99", "CI95", "CI90"), lwd = 2,
       col = c("#1B9E77", "#D95F02", "#7570B3", "#E7298A"),
       bg = "white")
box()
par(op)

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

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

Description

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

Usage

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

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

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

Arguments

`rwi`	a `data.frame` with detrended and standardized ring width indices as columns and years as rows such as that produced by `detrend`.
`ids`	an optional `data.frame` with column one named `"tree"` giving a `numeric` ID for each tree and column two named `"core"` giving a `numeric` ID for each core. Defaults to one core per tree as `data.frame(tree=1:ncol(`*rwi*`), core=rep(1, ncol(`*rwi*`)))`.

`period` a `character` string, either `"common"` or `"max"` indicating whether correlations should be limited to complete observations over the period common to all cores (i.e. rows common to all samples) or the maximum pairwise overlap. Defaults to `"max"`.

`method` Can be either `"pearson"`, `"kendall"`, or `"spearman"` which indicates the correlation coefficient to be used. Defaults to `"spearman"`. See `cor`.

`n` `NULL` or an integral value giving the filter length for the `hanning` filter used for removal of low frequency variation.

`nyrs` `NULL` or a number greater than zero. If not `NULL`, each series is divided by a smoothing spline with this rigidity (see `caps`) to remove low-frequency variation, as `detrend` does with `method = "Spline"`. A value of 1 or less is taken as a proportion of each series' length. Unlike the `hanning` filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.

`prewhiten` `logical` flag. If `TRUE` each series is whitened using `ar`.

`ar.order.max` `NULL` or a positive integer giving the maximum order of the `ar` model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If `NULL`, the order is chosen by AIC up to the `ar` default, which on long series can be 20 or more. Requires *prewhiten* = `TRUE`.

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

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

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

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

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

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

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

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

Details

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

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

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

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

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

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

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

Please note that Buras (2017) cautions against the use of expressed population signal (EPS) for evaluating the climate-reconstruction potential of a tree-ring sample. Instead, he recommends the use of subsample signal strength (`sss`) for evaluating the loss of predictive power back in time when sample replication drops.

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.

The statistics are correlations, so they do not depend on the level of each series. *rwi* may hold values that can be negative, such as indices from `detrend` with *difference* = `TRUE`, log widths or isotope values; each series then has its mean subtracted rather than being divided by it, which would flip a series with a negative mean. The *n* and *nyrs* filters divide each series by a smooth curve, so they refuse data with negative values.

Value

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

`start.year` the first year in the window. Not returned if `running.window` is `FALSE` or called as `rwi.stats` or `rwi.stats.legacy`.

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

Note

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

Author(s)

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

References

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

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

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

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

See Also

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

Examples

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

`rwl.check` *Check the integrity of a RWL object or a Tucson file*

Description

Run a battery of integrity checks against a `rwl` object or a Tucson format file and return the result as data: one row per finding, each carrying a stable check ID and a severity. Intended both for a researcher looking at one collection and for sweeping a large archive.

Usage

```
rwl.check(x, file = NULL,
          checks = c("structure", "series", "values", "zeros",
                     "crossdating", "provenance", "file"),
          control = rwl.check.control(), ...)

rwl.check.control(min.depth = 2, depth.run = 10, min.length = 30,
                  run.min = 6,
                  lag.max = 5, min.overlap = 50, spline.nyrs = 32,
                  r.dating = 0.35,
                  r.margin = 0.05, outlier.mad = 4, outlier.gap = 0.2,
                  r.cohesion = 0.35,
                  plausible.mean = c(0.05, 10),
                  small.thresh = NA, big.thresh = NA,
                  max.year = as.integer(format(Sys.Date(), "%Y")))

rwl.check.catalogue()
```

Arguments

`x`	a `rwl` object, an object coercible to one via `as.rwl`, or a length-one `character` giving the path to a Tucson format file, which is read with `read.tucson`.

file	a character naming the file the object came from. Used to label findings and to run the "file" checks. Taken from *x* when *x* is a path.
checks	a character vector of check groups to run. The "crossdating" group is much the most expensive; omit it for a fast pass over many files.
control	a list of tuning values, from rwl.check.control.
min.depth	years measured by fewer than this many series are flagged.
depth.run	how many consecutive such years are needed before they are reported. Every collection thins at its edges, so a year or two says nothing.
min.length	series with fewer than this many rings are flagged.
run.min	a run of this many identical non-zero rings is flagged.
lag.max	largest lag searched when looking for dating errors.
min.overlap	rings a series must share with the master to be checked against it.
spline.nyrs	stiffness, in years, of the spline used to high pass the series before the crossdating checks. Crossdating works on year-to-year variation, so correlating raw ring widths would measure the agreement of age trends instead. The default is the COFECHA value, which is also caps's own default and what corr.rwl.seg recommends.
r.dating	correlation a series must reach at its best lag before a non-zero lag is called a dating error.
r.margin	how much better than the correlation as dated that best-lag correlation must be.
outlier.mad	how many median absolute deviations below its own collection's median correlation a series must fall to be flagged as not fitting the collection.
outlier.gap	how far below the collection median, in correlation, that series must also fall. The second condition stops a very tight collection from flagging trivial spread.
r.cohesion	median series correlation below which the collection as a whole is reported as sharing little common signal.
plausible.mean	range, in mm, within which a collection's mean ring width is taken to be plausible.
small.thresh, big.thresh	thresholds for flagging small and large rings. NA omits the check.
max.year	years after this are flagged.
...	not used.

Details

rwl.check complements rwl.report. Where rwl.report describes a collection for a reader, rwl.check looks for defects and returns them as rows, so that examining one file and sweeping ten thousand are the same operation.

Every check runs independently and none of them stops. A check that fails becomes a finding with the ID RWL_CHECK_ERROR rather than an error, so a pathological file still yields a report. This matters at scale: the files that break are the ones worth looking at.

Each finding carries a stable check ID and one of three severities: "error" for something that cannot be right, "warning" for something probably wrong, and "note" for something worth knowing.

rwl.check.catalogue returns the full table of IDs, groups, severities and descriptions. The IDs are a contract: they are what a sweep filters on and what two runs are compared by, and they do not change meaning between releases.

The "provenance" checks read the record read.tucson attaches to what it returns, and report things that cannot be recovered any other way. Chief among them is RWL_ID_RENAMED: where a file gives two different cores the same id, the reader renames one, so a series on the returned object carries a name that appears nowhere in the file. Nothing but the reader can say that. The group is silent for an object that carries no such record, including one read by read.tucson.legacy.

The "file" checks need the file itself and are skipped when only an object is given. They are the only way to see line endings, stray tabs, and the span declared in the ITRDB header, all of which are gone once the file has been parsed into a matrix of numbers.

The crossdating checks ask whether a series fits the collection it is in rather than whether it clears a fixed correlation, because what counts as a good correlation is a property of the site: a high-elevation conifer stand runs 0.7 to 0.8, while some collections, particularly ecological ones sampled for growth rather than for a climate signal, sit far below that and are perfectly sound. A collection with little common signal is therefore reported once, as RWL_WEAK_COLLECTION and as a note rather than a warning: it is not a fault to be corrected, but it does decide whether the series can be crossdated against each other or carry a chronology, so it is worth saying.

The defaults in rwl.check.control were calibrated against the rwl objects shipped with dplR. In particular RWL_DATING_LAG fires on none of anos1, ca533, co021, gp.rwl, nm046 or wa082, and does fire on a series displaced by two years.

The value of an RWL_DATING_LAG finding is the lag at which the series correlates best with the master, with the sign used by ccf.series.rwl (with series.x = FALSE) and by best.lag in corr.rwl.seg: negative means the series is probably missing rings, positive that it probably has false rings or a ring counted twice.

Value

An object of class "rwl.check": a list with the file name, the control values used, a findings data.frame, and a meta list of summary values.

as.data.frame returns the findings, one row per finding, with columns file, check, severity, group, series, year.from, year.to, n, value and message.

summary returns a one-row data.frame holding the collection's summary values, counts of errors, warnings and notes, and one count column per check ID. Rows from different files have identical columns and can be combined with rbind into a triage table.

Author(s)

Andy Bunn

See Also

rwl.report, read.tucson, interseries.cor, corr.rwl.seg

Examples

```
library(utils)
data(zof.rwl)
res <- rwl.check(zof.rwl, file = "zof.rwl")
res

## the findings as data
head(as.data.frame(res))

## one row per file: rbind these over an archive to get a work queue
data(ca533)
sweep <- rbind(summary(rwl.check(zof.rwl, file = "zof.rwl")),
               summary(rwl.check(ca533,   file = "ca533")))
sweep[, c("file", "n.series", "n.error", "n.warning", "n.note")]

## what is looked for
head(rwl.check.catalogue())

## a fast pass, skipping the crossdating checks
rwl.check(ca533, checks = c("structure", "series", "values", "zeros"))
```

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

Description

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

Usage

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

Arguments

`rwl`	an `rwl` object of ring widths, such as produced by `read.rwl`. An `rwi` object of indices is reported on with a warning, since the report's means, zeros and thresholds are about widths. Anything else is an error.
`small.thresh`	a `numeric` value for the threshold value that will cause small rings to be listed. If values is `NA` this will be omitted.
`big.thresh`	a `numeric` value for the threshold value that will cause large rings to be listed. If values is `NA` this will be omitted.

Details

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

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

When the object still carries the record read.tucson attaches to what it returns, the printed report opens with a short header naming the file, the site and species from the ITRDB header, the precision the file declares, and the span the header claims, which is worth seeing beside the span the measurements actually cover. It also notes when the reader renamed a series, so the ids in the report are not the file's, and when interior gaps were filled. An object with no such record – one built by hand, read by read.tucson.legacy, or subsetted by column – is reported exactly as before. Use rwl.check to have any of this reported as findings rather than as context.

Value

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

small.thresh	a numeric value passed in.
big.thresh	a numeric value passed in.
nSeries	a numeric value with the number of series.
n	a numeric value with the total number of rings.
meanSegLength	a numeric with the mean segment length.
firstYear	a numeric with the first year.
lastYear	a numeric with the last year.
meanAR1	a numeric with the mean AR1 value of all series.
sdAR1	a numeric with the standard deviation of the AR1 values of all series.
unconnected	a logical indicating if there are rows (years) with all NA values.
unconnectedYrs	a numeric vector with unconnected years.
nZeros	a numeric with number of zeros in the data.
zeros	a list containing series and years with zeros or a numeric of 0.
allZeroYears	a numeric indicating rows (years) with all zero values.
consecutiveZeros	a list containing series and years with consecutive zeros or a numeric of 0.
meanInterSeriesCor	a numeric with the mean interseries correlation.
sdInterSeriesCor	a numeric with the standard deviation of the interseries correlations.
internalNAs	a list containing series and years with internal NA values or a numeric of 0.
provenance	a list of details taken from the record attached by read.tucson, or absent when the object carries no such record.
smallRings	a list containing series and years with small rings or a numeric of 0.
bigRings	a list containing series and years with large rings or a numeric of 0.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

read.rwl, summary.rwl, interseries.cor

Examples

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

---

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

Description

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

Usage

```
rwl.stats(rwl)

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

Arguments

rwl, object	a rwl object with (usually) raw ring-width series as columns and years as rows such as that produced by read.rwl. It is sometimes desirable to run this on detrended (e.g., rwi) data.
...	Additional arguments from the generic function. These are silently ignored.

Details

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

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

Value

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

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

Author(s)

Andy Bunn. Slightly improved by Mikko Korpela.

References

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

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

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

See Also

`rwi.stats`, `read.rwl`

Examples

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

---

sea *Superposed Epoch Analysis*

Description

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

Usage

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

Arguments

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

Details

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

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

Value

A data.frame with

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

Author(s)

Christian Zang. Patched and improved by Mikko Korpela.

References

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

Examples

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

---

seg.plot *Segment Plot*

Description

Makes a segment plot of tree-ring data.

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

spag.plot

Examples

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

sens1 *Calculate Mean Sensitivity*

Description

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

Usage

```
sens1(x)
```

Arguments

x	a `numeric` vector containing the series

Details

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

Value

the mean sensitivity.

Author(s)

Mikko Korpela, based on original by Andy Bunn

References

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

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

See Also

`sens2`, `rwl.stats`

Examples

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

sens2 *Calculate Mean Sensitivity on Series with a Trend*

Description

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

Usage

```
sens2(x)
```

Arguments

x	a `numeric` vector containing the series

Details

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

Value

the mean sensitivity.

Author(s)

Mikko Korpela, based on original by Andy Bunn

References

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

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

See Also

`sens1`, `rwl.stats`

Examples

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

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

Description

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

Usage

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

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)). Ignored if *series* is an index to a column of *rwl*.
seg.length	an even integral value giving length of segments in years (e.g., 20, 50, 100 years).
bin.floor	a non-negative integral value giving the base for locating the first segment (e.g., 1600, 1700, 1800 AD). Typically 0, 10, 50, 100, etc.
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
nyrs	NULL or a number greater than zero. If not NULL, each series is divided by a smoothing spline with this rigidity (see caps) to remove low-frequency variation, as detrend does with method = "Spline". A value of 1 or less is taken as a proportion of each series' length. Unlike the hanning filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
prewhiten	logical flag. If TRUE each series is whitened using ar.
ar.order.max	NULL or a positive integer giving the maximum order of the ar model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If NULL, the order is chosen by AIC up to the ar default, which on long series can be 20 or more. Requires *prewhiten* = TRUE.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
floor.plus1	logical flag. If TRUE, one year is added to the base location of the first segment (e.g., 1601, 1701, 1801 AD).

Details

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

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

Plot 2 Scatterplot of series vs. master

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

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

The series and master are returned as well.

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

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

Examples

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

sgc *Synchronous Growth Changes*

Description

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

Usage

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

Arguments

rwl	an rwl object such as that produced by read.rwl. Also accepts a data.frame with series in columns and years as row names.
overlap	integer value with minimal length of overlapping growth changes (compared number of tree rings - 1). Comparisons with less overlap are not compared.
prob	if TRUE then the probability of exceedence of the sgc will be calculated

Details

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

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

Value

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

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

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

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

Author(s)

Ronald Visser

References

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

See Also

glk

Examples

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

---

skel.plot *Skeleton Plot*

Description

Automatically generates a skeleton plot of tree-ring data.

Usage

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

Arguments

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

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

Details

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

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

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

Stokes, M. A. and Smiley, T. L. (1968) *An Introduction to Tree-Ring Dating*. The University of Arizona Press. ISBN-13: 978-0-8165-1680-3.

Sheppard, P. R. (2002) Crossdating Tree Rings Using Skeleton Plotting. https://www.ltrr.arizona.edu/skeletonplot/introcrossdate.htm.

Meko, D. (2002) Tree-Ring MATLAB Toolbox. https://dmeko.ltrr.arizona.edu/toolbox.html.

See Also

Devices, hanning, xskel.plot, xskel.ccf.plot

Examples

```
library(utils)
data(co021)
x <- co021[,33]
x.yrs <- time(co021)
x.name <- colnames(co021)[33]
## On a raw ring width series - undated
skel.plot(x)
## On a raw ring width series - dated with names
skel.plot(x, yr.vec = x.yrs, sname = x.name, master = TRUE)
## Not run:
library(grDevices)
## Try cross-dating
y <- co021[, 11]
y.yrs <- time(co021)
y.name <- colnames(co021)[11]

## send to postscript - 3 pages total
fname1 <- tempfile(fileext=".ps")
print(fname1) # tempfile used for PS output
postscript(fname1)
## "Master series" with correct calendar dates
skel.plot(x, yr.vec = x.yrs, sname = x.name, master = TRUE)
## Undated series, try to align with last plot
skel.plot(y)
## Here's the answer...
skel.plot(y, yr.vec = y.yrs, sname = y.name)
dev.off()

unlink(fname1) # remove the PS file

## alternatively send to pdf
fname2 <- tempfile(fileext=".pdf")
print(fname2) # tempfile used for PDF output
pdf(fname2, width = 10, height = 7.5, paper = "USr")
skel.plot(x, yr.vec = x.yrs, sname = x.name, master = TRUE)
skel.plot(y)
skel.plot(y, yr.vec = y.yrs, sname = y.name)
dev.off()

unlink(fname2) # remove the PDF file

## End(Not run)
```

---

spag.plot *Spaghetti Plot*

Description

Makes a spaghetti plot of tree-ring data.

Usage

```
spag.plot(rwl, zfac = 1, useRaster = FALSE, res = 150, ...)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
zfac	a multiplier for *rwl* to enhance clarity when plotting.
useRaster	A logical flag. If TRUE, the tree-ring series are drawn as a raster image. Other parts of the plot are not affected. Other choices are NA (automatic) and FALSE (use vector graphics, the default). See wavelet.plot.
res	A numeric vector of length 1. The resolution (pixels per inch) of the tree-ring series when a raster image is used.
...	arguments to be passed to lines.

Details

This makes a simple plot of each series in a tree-ring data set. Each series is centered first by subtracting the column mean using scale. The plot can be grossly tuned with *zfac* which is a multiplier to *rwl* before plotting and centering.

Ring-width indices (class "rwi", see as.rwi) are centred on 1, or on 0 if they are differences, and not on their means, so the grey line under each series is the value the indices should sit at.

Value

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

seg.plot

Examples

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

ssf *Simple Signal Free Standardization*

Description

A simple implementation of the signal-free chronology

Usage

```
ssf(rwl,
    method="Spline",
    nyrs = NULL,
    difference = FALSE,
    max.iterations = 25,
    mad.threshold = 5e-4,
    recode.zeros = FALSE,
    return.info = FALSE,
    verbose = TRUE)
```

Arguments

`rwl`	a `rwl` object with series as columns and years as rows such as that produced by `read.rwl`. See 'Details'.
`method`	a `character` vector to determine the detrending method. See 'Details' below. Possible values are `c("Spline", "AgeDepSpline")`.
`nyrs`	a number controlling the smoothness of the fitted curve in methods. See 'Details' in `detrend.series`.
`difference`	a `logical` flag. Compute residuals by subtraction if TRUE, otherwise use division. See 'Details' in `detrend.series`
`max.iterations`	a `numeric`. The maximum number of iterations to be used in the fitting process. See 'Details'.
`mad.threshold`	a `numeric`. The median absolute difference threshold used for a stopping criteria. See 'Details'.
`recode.zeros`	a `logical` flag. Replace values of zero in `rwl` with 0.001 if TRUE.
`return.info`	a `logical` flag. If `TRUE`, details about models and data will be added to the return value. See 'Value'.
`verbose`	a `logical` flag. Write out details to the screen?

Details

This function creates a simple signal-free chronology that loosely follows the procedures laid out on p75 of Melvin and Briffa (2008). This function is a lighter version of that procedure and users who want more control and refinement should look to the CRUST program described in Melvin and Briffa (2014). These steps are described in more detail in Learning to Love dplR.

1. Detrend each series using the selected method, calculate RWI by division (or subtraction), and create an initial mean-value chronology.
2. Create signal-free measurements by dividing (or subtracting) each series of measurements by the chronology. If `return.info` is invoked these are returned in `sfRW_Array`.
3. Rescale the signal-free measurements to their original mean. If `return.info` is invoked these are returned in `sfRWRescaled_Array`.
4. If the sample depth is one, replace signal-free measurements with original measurements.
5. Fit curves to signal free measurements.If `return.info` is invoked these are returned in `sfRWRescaledCurves_Array`.
6. Get new growth indicies by dividing (or subtracting) the original measurements by curves in the last step. If `return.info` is invoked these are returned in `sfRWI_Array`.
7. Create a mean-value chronology using the indicies from the prior step. If `return.info` is invoked these are returned in `sfCrn_Mat`.
8. Repeat steps two through seven up to `maxIter` or until the `madThreshold` is reached. The stopping criteria is determined using the absolute difference between filtered chronologies generated in interation k and k-1. This is done with the residuals of a high-pass filter on the chronology using a cubic smoothing spline (`caps`) with the stiffness set as the median of the segment lengths of series contributing to the chronology. The stopping threshold is calculated as the median absolute difference of the kth and kth-1 chronologies weighted by the normalized sample depth. If `return.info` is invoked the residual chronologies are returned in `hfCrnResids_Mat` and the median absolute differences are returns in `MAD_Vec`.

The input object (`rwl`) should be of `class` `rwl`. If it not, the function will attempt to coerce it using `as.rwl` and a warning will be issued.

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

Value

An object of of class `crn` and `data.frame` with the signal-free chronology and the sample depth. The years are stored as row numbers.

If `return.info` is `TRUE` a `list` containing ouptput by iteration (k):

`infoList` a `list` with information on the arguments used in the function call.

`k` a `numeric` containing the number of iterations.

`ssfCrn` the signal-free chronology as above.

`sfRW_Array` an `array` of years by series by iteration (0:k) that holds the signal free measurements. The values in `sfRW_Array[,,1]` are what is used at the start of the iterative process (iteration 0) which here are the original data.

`sfRWRescaled_Array` an `array` of years by series by iteration (0:k) that holds the rescaled signal free measurements. The values in `sfRWRescaled_Array[,,1]` are what is used at the start of the iterative process (iteration 0) which here are the original data.

`sfRWRescaledCurves_Array` an `array` of years by series by iteration (0:k) that holds the rescaled signal free curve fits. The values in `sfRWRescaledCurves_Array[,,1]` are the curves used at the start of the iterative process (iteration 0).

sfRWI_Array	an array of years by series by iteration (0:k) that holds the detrended signal free measurements. The values in sfRWI_Array[,,1] are the RWI values at the start of the iterative process (iteration 0).
sfCrn_Mat	a matrix of years by iteration (0:k) that holds the calculated chronology. The values in sfCrn_Mat[,1] is the initial chronology from the start of the iterative process (iteration 0).
hfCrn_Mat	a matrix of years by iteration (0:k) that holds the calculated high-frequency chronology. The values in hfCrn_Mat[,1] is the initial high-frequency chronology from the start of the iterative process (iteration 0).
hfCrnResids_Mat	a matrix of years by k-1 that holds the differences between the kth and the kth-1 high frequency chronology residuals.
MAD_out	a vector containing the median absolute difference between iteration k and k-1.

Author(s)

Ed Cook provided Fortran code that was ported to R by Andy Bunn.

References

Melvin, TM, Briffa, KR (2008) A 'signal-free' approach to dendroclimatic standardisation. Dendrochronologia 26: 71–86 doi: 10.1016/j.dendro.2007.12.001

Melvin T. M. and Briffa K.R. (2014a) CRUST: Software for the implementation of Regional Chronology Standardisation: Part 1. Signal-Free RCS. Dendrochronologia 32, 7-20, doi: 10.1016/j.dendro.2013.06.002

Melvin T. M. and Briffa K.R. (2014b) CRUST: Software for the implementation of Regional Chronology Standardisation: Part 2. Further RCS options and recommendations. Dendrochronologia 32, 343-356, doi: 10.1016/j.dendro.2014.07.008

See Also

detrend. chron

Examples

```
library(stats)
data(wa082)
wa082_SSF <- ssf(wa082)
plot(wa082_SSF,add.spline=TRUE,nyrs=20)

wa082_SSF_Full <- ssf(wa082,method = "AgeDepSpline",
  difference = TRUE, return.info = TRUE)
plot(wa082_SSF_Full$ssfCrn,add.spline=TRUE,nyrs=20)
```

sss *Subsample Signal Strength*

Description

Calculate subsample signal strength on a `data.frame` of (usually) ring-width indices.

Usage

```
sss(rwi, ids = NULL)
```

Arguments

`rwi`	a `data.frame` with detrended and standardized ring width indices as columns and years as rows such as that produced by `detrend`.
`ids`	an optional `data.frame` with column one named `"tree"` giving a `numeric` ID for each tree and column two named `"core"` giving a `numeric` ID for each core. Defaults to one core per tree as `data.frame(tree=1:ncol(rwi), core=rep(1, ncol(rwi)))`.

Details

This calculates subsample signal strength (sss) following equation 3.50 in Cook and Kairiukstis (1990) but using notation from Buras (2017) because writing the prime unicode symbol seems too difficult. The function calls `rwi.stats` and passes it the arguments `ids` and `prewhiten`.

To make better use of variation in growth within and between series, an appropriate mask (parameter *ids*) should be provided that identifies each series with a tree as it is common for dendrochronologists to take more than one core per tree. The function `read.ids` is helpful for creating a mask based on the series ID.

Subsample signal strength is calculated as $\frac{n[1+(N-1)\bar{r}]}{N[1+(n-1)\bar{r}]}$ where `n` and `N` are the number of cores or trees in the subsample and sample respectively and `rbar` is mean interseries correlation. If there is only one core per tree `n` is the sample depth in a given year (`rowSums(!is.na(rwi))`), `N` is the number of cores (`n.cores` as given by `rwi.stats`), and `rbar` is the mean interseries correlation between all series (`r.bt` as given by `rwi.stats`). If there are multiple cores per tree `n` is the number of trees present in a given year, `N` is the number of trees (`n.trees` as given by `rwi.stats`), and `rbar` is the effective mean interseries correlation (`r.eff` as given by `rwi.stats`).

Readers interested in the differences between subsample signal strength and the more commonly used (running) expressed population signal should look at Buras (2017) on the frequently mis-used citation of Wigley et al. (1984) of the expressed population signal threshold EPS=0.85 as well as Cook and Pederson (2011) for a more general approach to categorizing variability in tree-ring data.

Value

A `numeric` containing the subsample signal strength that is the same as number if rows of`rwi`.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

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

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

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

Wigley, T. M., Briffa, K. R. and Jones, P. D. (1984) On the average value of correlated time series, with applications in dendroclimatology and hydrometeorology. Journal of Applied Meteorology and Climatology 23: 201-213.

See Also

rwi.stats, read.ids

Examples

```
data(ca533)
ca533.rwi <- detrend(ca533,method="Spline")
# assuming 1 core / tree
ca533.sss <- sss(ca533.rwi)

ca533.ids <- autoread.ids(ca533)
# done properly with >=1 core / tree as per the ids
ca533.sss2 <- sss(ca533.rwi,ca533.ids)

yr <- time(ca533)
plot(yr,ca533.sss,type="l",ylim=c(0.4,1),
     col="darkblue",lwd=2,xlab="Year",ylab="SSS")
lines(yr,ca533.sss2,lty="dashed",
      col="darkgreen",lwd=2)

# Plot the chronology showing a potential cutoff year based on SSS
# (using sss2 with the correct series IDs to get >=1 core / tree as per the ids)
ca533.crn <- chron(ca533.rwi)
def.par <- par(no.readonly=TRUE)
par(mar = c(2, 2, 2, 2), mgp = c(1.1, 0.1, 0), tcl = 0.25, xaxs='i')
plot(yr, ca533.crn[, 1], type = "n", xlab = "Year",
     ylab = "RWI", axes=FALSE)
cutoff <- max(yr[ca533.sss2 < 0.85])
xx <- c(500, 500, cutoff, cutoff)
yy <- c(-1, 3, 3, -1)
polygon(xx, yy, col = "grey80")
abline(h = 1, lwd = 1.5)
lines(yr, ca533.crn[, 1], col = "grey50")
lines(yr, caps(ca533.crn[, 1], nyrs = 32), col = "red", lwd = 2)
```

```
axis(1); axis(2); axis(3);
par(new = TRUE)
## Add SSS
plot(yr, ca533.sss2, type = "l", xlab = "", ylab = "",
     axes = FALSE, col = "blue")
abline(h=0.85,col="blue",lty="dashed")
axis(4, at = pretty(ca533.sss2))
mtext("SSS", side = 4, line = 1.1, lwd=1.5)
box()
par(def.par)
```

`strip.rwl` *Chronology Stripping by* EPS

Description

EPS-based chronology stripping after Fowler & Boswijk 2003.

Usage

```
strip.rwl(rwl, ids = NULL, verbose = FALSE, comp.plot = FALSE,
          legacy.eps = FALSE)
```

Arguments

`rwl`	a `data.frame` of raw tree-ring widths series, such as that produced by `read.rwl` or `read.fh`
`ids`	an optional `data.frame` with column one named `"tree"` giving a `numeric` ID for each tree and column two named `"core"` giving a `numeric` ID for each core. This is passed on to `rwi.stats`. See its manual for the meaning of the default value `NULL` and more information.
`verbose`	`logical` flag, indicating if the EPS calculated at each step and other details should be printed on the console during the chronology stripping process
`comp.plot`	`logical` flag, indicating if a diagnostic plot with year-wise stripped and unstripped EPS should be drawn (see details below)
`legacy.eps`	`logical` flag, indicating if the EPS will be calculated with `rwi.stats` (`FALSE`, the default) or `rwi.stats.legacy` (`TRUE`)

Details

The EPS-based chronology stripping is implemented after Fowler & Boswijk 2003: First, all series are standardized using a double detrending procedure with splines and frequency cutoffs of 50% at 20 and 200 years. Then, EPS is calculated for the chronology including all (remaining) series. In each iteration, the algorithm calculates leave-one-out EPS values, and the series whose removal increases overall EPS the most is discarded. This is repeated until no further increase in EPS is gained by discarding a single series. The procedure is then repeated in the opposite direction,

i.e., the reinsertion of each previously removed series into the `data.frame` is considered. In each iteration, the series (if any) whose reinsertion increases EPS the most is reinserted. As a last step, EPS is calculated for each year of the stripped and original chronology including all series. If `comp.plot` is set to `TRUE`, a diagnostic plot is shown for the year-wise comparison.

When verbose output is chosen, the EPS values for all leave-one-out (or back-in) chronologies are reported. If discarding or re-inserting a single series leads to an improvement in EPS, this series is marked with an asterisk.

Value

The functions returns a `data.frame` of raw tree-ring widths, where series that do not contribute to an overall improvement in EPS are left out.

Author(s)

Christian Zang. Patched and improved by Mikko Korpela.

References

Fowler, A. and Boswijk, G. (2003) Chronology stripping as a tool for enhancing the statistical quality of tree-ring chronologies. *Tree-Ring Research*, **59**(2), 53–62.

See Also

`rwi.stats`

Examples

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

---

tbrm *Calculate Tukey's Biweight Robust Mean*

Description

This calculates a robust average that is unaffected by outliers.

Usage

```
tbrm(x, C = 9)
```

Arguments

x	a numeric vector
C	a constant. *C* is preassigned a value of 9 according to the Cook reference below but other values are possible.

Details

This is a one step computation that follows the Affy whitepaper below, see page 22. This function is called by chron to calculate a robust mean. *C* determines the point at which outliers are given a weight of 0 and therefore do not contribute to the calculation of the mean. *C* = 9 sets values roughly +/-6 standard deviations to 0. *C* = 6 is also used in tree-ring chronology development. Cook and Kairiukstis (1990) have further details.

An exact summation algorithm (Shewchuk 1997) is used. When some assumptions about the rounding of floating point numbers and conservative compiler optimizations hold, summation error is completely avoided. Whether the assumptions hold depends on the platform, i.e. compiler and CPU.

Value

A numeric mean.

Author(s)

Mikko Korpela

References

Statistical Algorithms Description Document, 2002, Affymetrix.

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

Mosteller, F. and Tukey, J. W. (1977) *Data Analysis and Regression: a second course in statistics*. Addison-Wesley. ISBN-13: 978-0-201-04854-4.

Shewchuk, J. R. (1997) Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete and Computational Geometry*, **18**(3), 305–363.

See Also

chron

Examples

```
library(stats)
library(utils)
foo <- rnorm(100)
tbrm(foo)
mean(foo)

## Compare
```

```
data(co021)
co021.rwi <- detrend(co021, method = "ModNegExp")
crn1 <- apply(co021.rwi, 1, tbrm)
crn2 <- chron(co021.rwi)
cor(crn1, crn2[, 1])
```

`time.rwl` *Retrieve or set the time values for rwl and crn objects*

Description

Retrieve or set the time values for rwl and crn objects.

Usage

```
## S3 method for class 'rwl'
time(x, ...)
## S3 method for class 'crn'
time(x, ...)
time(x) <- value
```

Arguments

`x`	An object of class `"rwl"` or an aboject of class `"crn"`.
`...`	Not used.
`value`	A `numeric` vector to assign the time.

Value

A `numeric` vector of time (typically in years) for the object. This is done via `as.numeric(rownames(x))` but has been asked for by users so many times that it is being included as a function.

Author(s)

Andy Bunn

See Also

`read.rwl` `read.crn`

Examples

```
library(utils)
data(co021)
# extract years
co021.yrs <- time(co021)
# set years -- silly example
time(co021) <- co021.yrs+100
```

treeMean *Calculate mean across cores in a tree*

Description

This function calculates the mean value for each tree in a rwl or rwi object.

Usage

```
treeMean(rwl, ids, na.rm=FALSE)
```

Arguments

rwl	a data.frame of ring widths with rownames(*rwl*) containing years and colnames(rwl) containing each series ID such as produced by read.rwl
ids	a data.frame with column one named "tree" giving a numeric ID for each tree and column two named "core" giving a numeric ID for each core.
na.rm	logical passed to rowMeans. Should missing values be removed?

Details

This function averages together multiple cores to give a mean value of growth. It is very common in dendrochronology to take more than one core per tree. In those cases it is occasionally desirable to have an average of the cores. This function merely loops through the rwl object and calculates the rowMeans for each tree. If na.rm=TRUE trees with >1 sample will be averaged only over the period where the samples overlap. If FALSE the output can vary in the number of samples. See examples.

Value

An object of class c("rwl", "data.frame") with the mean annual value for each tree.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

read.rwl, read.ids

Examples

```
data(gp.rwl)
gp.ids <- read.ids(gp.rwl, stc = c(0, 2, 1))

gp.treeMean <- treeMean(gp.rwl, gp.ids)
gp.treeMean2 <- treeMean(gp.rwl, gp.ids, na.rm=TRUE)

# look at an example of a single tree with different averaging periods.
```

```
# The years are named on every subset so that the cores and the two tree
# means stay on one grid: selecting series alone would give each of them
# only the years it covers, and the three spans are not the same.
yrs <- rownames(gp.rwl)
tree40 <- data.frame(gp.rwl[yrs, c("40A","40B")],
                     gp.treeMean[yrs, "40", drop=FALSE],
                     gp.treeMean2[yrs, "40", drop=FALSE])
names(tree40) <- c("coreA", "coreB", "treeMean1", "treeMean2")
head(tree40,50)

data(ca533)
ca533.treeMean <- treeMean(ca533, autoread.ids(ca533))
# plot using S3method for class "rwl"
plot(ca533.treeMean,plot.type="spag")
```

`tridas.vocabulary` *Browse and Check Standard TRiDaS Vocabulary*

Description

This function can be used to browse the TRiDaS vocabulary by category.

Usage

```
tridas.vocabulary(category = c("dating type", "measuring method",
                      "shape", "location type", "variable", "unit",
                      "remark", "dating suffix", "presence / absence",
                      "complex presence / absence", "certainty"),
                  idx = NA, term = NA, match.exact = FALSE)
```

Arguments

`category`	Vocabulary category as a `character` vector of length one. One of `"dating type"`, `"measuring method"`, `"shape"`, `"location type"`, `"variable"`, `"unit"`, `"remark"`, `"dating suffix"`, `"presence / absence"`, `"complex presence / absence"`, `"certainty"`. Partial matches are allowed.
`idx`	A `numeric` vector. Index to the `character` vector containing the vocabulary of the given category.
`term`	A `character` vector. One or more (partial) terms to look for in the given category.
`match.exact`	A `logical` value. If `TRUE`, partial matching of *term* is not used. Defaults to `FALSE`.

Details

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

The function has four usage modes:

1. When *idx* is given, returns item number *idx* in the given *category*. There may be several numbers in *idx*, in which case multiple items are returned.
2. When *term* contains one or more items and *match.exact* is `TRUE`, checks whether any of the terms is an exact match in the given *category*
3. When *term* contains one or more items and *match.exact* is `FALSE`, expands partial matches of the terms in the vocabulary of the given *category*
4. When only *category* is given, returns the complete vocabulary in the given *category*

Value

In mode 1	A `character` vector, same length as in *idx*
In mode 2	A `logical` value
In mode 3	A `character` vector, same length as in *term*
In mode 4	A `character` vector

Author(s)

Mikko Korpela

References

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

See Also

`read.tridas`, `write.tridas`

Examples

```
## Show all entries in category "measuring method"
tridas.vocabulary(category = "measuring")

## Show item number one in category "complex presence / absence"
tridas.vocabulary(category = "complex", idx = 1)

## Check whether "half section" exists in category "shape"
tridas.vocabulary(category = "shape", term = "half section",
                  match.exact = TRUE)

## Return unabbreviated matches to several queries in category "remark"
tridas.vocabulary(category = "remark",
                  term = c("trauma", "fire", "diffuse"))
```

uuid.gen UUID *Generator*

Description

Initializes and returns a generator of universally unique identifiers. Use the returned function repeatedly for creating one or more UUIDs, one per function call.

Usage

```
uuid.gen(more.state = "")
```

Arguments

more.state	A character string for altering the state of the generator

Details

This function returns a function (closure) which generates UUIDs. The state of that anonymous function is set when uuid.gen is called. The state consists of the following:

- System and user information (Sys.info)
- R version (R.version)
- Platform information (.Platform)
- Working directory
- Process ID of the R session
- Time when uuid.gen was called (precision of seconds or finer)
- The text in parameter *more.state*

The Pseudo Random Number Generator of R (see .Random.seed) is used in the generation of UUIDs. No initialization of the PRNG is done. Tampering with the state of the R PRNG while using a given UUID generator causes a risk of non-unique identifiers. Particularly, setting the state of the PRNG to the same value before two calls to the UUID generator guarantees two identical identifiers. If two UUID generators have a different state, it is *not* a problem to have the PRNG going through or starting from the same state with both generators.

The user is responsible for selecting a PRNG with a reasonable number of randomness. Usually, this doesn't require any action. For example, any PRNG algorithm available in R works fine. However, the uniqueness of UUIDs can be destroyed by using a bad user-supplied PRNG.

The UUIDs produced by uuid.gen generators are Version 4 (random) with 122 random bits and 6 fixed bits. The UUID is presented as a character string of 32 hexadecimal digits and 4 hyphens:

'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'

where *x* is any hexadecimal digit and *y* is one of "8", "9", "a", or "b". Each *x* and *y* in the example is an independent variables (for all practical purposes); subscripts are omitted for clarity. The UUID generator gets 32 hex digits from the MD5 message digest algorithm by feeding it a string consisting of the constant generator state and 5 (pseudo) random numbers. After that, the 6 bits are fixed and the hyphens are added to form the final UUID.

Value

A parameterless function which returns a single UUID (character string)

Author(s)

Mikko Korpela

References

Leach, P., Mealling, M., and Salz, R. (2005) A Universally Unique IDentifier (UUID) URN namespace. RFC 4122, RFC Editor. https://www.rfc-editor.org/rfc/rfc4122.txt.

See Also

digest, Random

Examples

```
## Normal use
ug <- uuid.gen()
uuids <- character(100)
for(i in 1:100){
  uuids[i] <- ug()
}
length(unique(uuids)) == 100 # TRUE, UUIDs are unique with high probability

## Do NOT do the following unless you want non-unique IDs
rs <- .Random.seed
set.seed(0L)
id1 <- ug()
set.seed(0L)
id2 <- ug()
id1 != id2 # FALSE, The UUIDs are the same
.Random.seed <- rs

## Strange usage pattern, but will probably produce unique IDs
ug1 <- uuid.gen("1")
set.seed(0L)
id1 <- ug1()
ug2 <- uuid.gen("2")
set.seed(0L)
id2 <- ug2()
id1 != id2 # TRUE, The UUIDs are different with high probability
.Random.seed <- rs
```

wa082 *Hurricane Ridge, Pacific silver fir*

Description

This data set gives the raw ring widths for Pacific silver fir *Abies amabilis* at Hurricane Ridge in Washington, USA. There are 23 series. Data set was created using `read.rwl` and saved to an .rda file using `save`.

The source file records no measurement for series `712011` in the year 1900, holding the value `-999` there. This data set was built before `read.tucson` distinguished a missing measurement from a ring width of zero, and that cell holds a zero. It has been left as it is for continuity with earlier versions of dplR. Reading the source file today gives `NA` in that cell; passing `fill.internal.NA = 0` to `read.tucson` reproduces this data set exactly.

Usage

```
data(wa082)
```

Format

A `data.frame` containing 23 tree-ring series in columns and 286 years in rows.

Source

International tree-ring data bank, Accessed on 20-April-2021 at https://www.ncei.noaa.gov/pub/data/paleo/treering/measurements/northamerica/usa/wa082.rwl

References

Schweingruber, F. (1983) Hurricane Ridge Data Set. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series 1983-wa082.RWL. NOAA/NCDC Paleoclimatology Program, Boulder, Colorado, USA.

Examples

```
library(utils)
data(wa082)

## Where the data came from. read.tucson() records what it saw and attaches
## it to what it returns; this data set carries that record.
prov <- attr(wa082, "dplR.provenance")
prov$file
prov$header

## The file records no measurement for series 712011 in 1900, holding -999
## there. This data set was built with fill.internal.NA = 0, so that cell
## holds a zero. See the description above.
prov$gaps
```

```
wa082["1900", "712011"]

## rwl.report() shows the same in its header
rwl.report(wa082)
```

`wavelet.plot` *Plot a Continuous Wavelet Transform*

Description

This function creates a `filled.contour` plot of a continuous wavelet transform as output from `morlet`.

Usage

```
wavelet.plot(wave.list,
             wavelet.levels = quantile(wave.list$Power,
                                       probs = (0:10)/10),
             add.coi = TRUE, add.sig = TRUE,
             x.lab = gettext("Time", domain = "R-dplR"),
             period.lab = gettext("Period", domain = "R-dplR"),
             crn.lab = gettext("RWI", domain = "R-dplR"),
             key.cols = rev(rainbow(length(wavelet.levels)-1)),
             key.lab = parse(text=paste0("\"",
                                         gettext("Power",
                                                 domain="R-dplR"),
                                         "\"^2")),
             add.spline = FALSE, f = 0.5, nyrs = NULL,
             crn.col = "black", crn.lwd = 1,coi.col='black',
             crn.ylim = range(wave.list$y) * c(0.95, 1.05),
             side.by.side = FALSE,
             useRaster = FALSE, res = 150, reverse.y = FALSE, ...)
```

Arguments

`wave.list`	A `list`. Output from `morlet`.
`wavelet.levels`	A `numeric` vector. Values for levels of the filled contours for the wavelet plot.
`add.coi`	A `logical` flag. Add cone of influence?
`add.sig`	A `logical` flag. Add contour lines for significance?
`x.lab`	X-axis label.
`period.lab`	Y-axis label for the wavelet plot.
`crn.lab`	Y-axis label for the time-series plot.
`key.cols`	A vector of colors for the wavelets and the key.
`key.lab`	Label for key.
`add.spline`	A `logical` flag. Add a spline to the time-series plot using `caps`?.

nyrs	A number giving the rigidity of the smoothing spline, defaults to 0.33 of series length if *nyrs* is NULL.
f	A number between 0 and 1 giving the frequency response or wavelength cutoff for the smoothing spline. Defaults to 0.5.
crn.col	Line color for the time-series plot.
crn.lwd	Line width for the time-series plot.
coi.col	Color for the COI if *add.coi* is TRUE.
crn.ylim	Axis limits for the time-series plot.
side.by.side	A logical flag. Plots will be in one row if TRUE.
useRaster	A logical flag. If TRUE, the filled contours are drawn as a raster image. Other parts of the plot are not affected. useRaster=TRUE can be especially useful when a pdf device is used: the size and complexity of the PDF file will probably be greatly reduced. Setting this to TRUE has negative effects when used with a bitmap device such as png. If NA, plotting of a raster image will be attempted if and only if the name of the graphics device is "pdf" or "postscript". The default is FALSE: draw directly to the graphics device without using an intermediate raster image.
res	A numeric vector of length 1. The resolution (pixels per inch) of the filled contours when a raster image is used. See *useRaster*.
reverse.y	A logical flag. If TRUE, the Y-axis will be reversed, i.e. period increasing towards the bottom. The default is FALSE.
...	Arguments passed to rasterPlot. Only relevant when the filled contours are drawn as a raster image. See *useRaster*.

Details

This produces a plot of a continuous wavelet transform and plots the original time series. Contours are added for significance and a cone of influence polygon can be added as well. Anything within the cone of influence should not be interpreted.

The time series can be plotted with a smoothing spline as well.

Value

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

Note

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

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

References

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

See Also

morlet, caps

Examples

```
library(stats)
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi, prewhiten = FALSE)
Years <- time(ca533.crn)
CAMstd <- ca533.crn[, 1]
out.wave <- morlet(y1 = CAMstd, x1 = Years, p2 = 9, dj = 0.1,
                   siglvl = 0.99)
wavelet.plot(out.wave, useRaster = NA)
## Not run:
# Alternative palette with better separation of colors
# via: rev(RColorBrewer::brewer.pal(10, "Spectral"))
specCols <- c("#5E4FA2", "#3288BD", "#66C2A5", "#ABDDA4", "#E6F598",
              "#FEE08B", "#FDAE61", "#F46D43", "#D53E4F", "#9E0142")
wavelet.plot(out.wave, key.cols=specCols,useRaster = NA)

# fewer colors
levs <- quantile(out.wave$Power, probs = c(0, 0.5, 0.75, 0.9, 0.99))
wavelet.plot(out.wave, wavelet.levels = levs, add.sig = FALSE,
             key.cols = c("#FFFFFF", "#ABDDA4", "#FDAE61", "#D7191C"), useRaster = NA)

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

wc	A data.frame containing wood completeness data, as returned by read.tridas.

Details

Computes the sum of the variables *n.missing.heartwood* and *n.unmeasured.inner* in *wc*.

Value

A `data.frame` containing two variables. Variable one (*series*) gives the series ID as either `characters` or `factors`. These match `rownames(wc)`. Variable two (*pith.offset*) is of `integer` type and gives the years from the beginning of the core to the pith (or center) of the tree. The minimum value is 1.

Author(s)

Mikko Korpela

See Also

`po.to.wc`, `rcs`, `read.tridas`

Examples

```
library(utils)
data(gp.po)
all(wc.to.po(po.to.wc(gp.po)) == gp.po)
```

`window.rwl` *Take a Span of Years*

Description

Take the years from `start` to `end` of an `"rwl"`, `"rwi"`, `"bai"` or `"crn"` object.

Usage

```
## S3 method for class 'rwl'
window(x, start = NULL, end = NULL, ...)

## S3 method for class 'rwi'
window(x, start = NULL, end = NULL, ...)

## S3 method for class 'bai'
window(x, start = NULL, end = NULL, ...)

## S3 method for class 'crn'
window(x, start = NULL, end = NULL, ...)
```

Arguments

x	an object of class "rwl", "rwi", "bai" or "crn".
start, end	the first and last years to keep, each a single number. NULL means the first or last year of x.
...	not used.

Details

The years are those in time(x). The rows are taken with [, so an "rwl", "rwi" or "bai" object keeps its class and its provenance record, cut down to the years that remain (see [.rwl). Series with no values in the window are dropped from an "rwl", "rwi" or "bai" object, with a message naming them, since an empty series is not one any dplR function can use; if no series has values in the window, that is an error. The years returned are still the ones asked for. subset drops empty series in the same way. x[i,] does not: [returns every series it is asked for, empty or not, so that its columns stay lined up with anything indexed alongside them (see [.rwl). Use window to take years for analysis.

A window is usually taken to line the data up with something else, so a window that runs past either end of x gives a warning that says which years were returned, and a window that does not overlap x at all is an error.

Value

An object of the same class as x, holding the years from start to end and the series with values in them.

Author(s)

Andy Bunn

See Also

time.rwl, [.rwl, window

Examples

```
data(ca533)
ca533.1800s <- window(ca533, 1800, 1899)
range(time(ca533.1800s))
ca533.crn <- chron(detrend(ca533, method = "Spline"))
tail(window(ca533.crn, start = 1950))
```

write.compact *Write* DPL *Compact Format Ring Width File*

Description

This function writes a chronology to a DPL compact format file.

Usage

```
write.compact(rwl.df, fname, append = FALSE, prec = 0.01,
              mapping.fname = "", mapping.append = FALSE, ...)
```

Arguments

rwl.df	a data.frame containing tree-ring ring widths with the series in columns and the years as rows. The series IDs are the column names and the years are the row names. This type of data.frame is produced by read.rwl and read.compact.
fname	a character vector giving the file name of the rwl file.
append	logical flag indicating whether to append this chronology to an existing file.
prec	numeric indicating the precision of the output file. This must be equal to either 0.01 or 0.001 (units are in mm).
mapping.fname	a character vector of length one giving the file name of an optional output file showing the mapping between input and output series IDs. The mapping is only printed for those IDs that are transformed. An empty name (the default) disables output.
mapping.append	logical flag indicating whether to append the description of the altered series IDs to an existing file. The default is to create a new file.
...	Unknown arguments are accepted but not used.

Details

The output should be readable by the Dendrochronology Program Library (DPL) as a compact format file.

In series IDs, letters of the English alphabet and numbers are allowed. Other characters will be removed. The length of the IDs is limited to about 50 characters, depending on the length of the other items to be placed on the header lines of the output file. Longer IDs will be truncated. Also any duplicate IDs will be automatically edited so that only unique IDs exist. If series IDs are changed, one or more warnings are shown. In that case, the user may wish to print a list of the renamings (see Arguments).

Value

fname

Author(s)

Mikko Korpela, based on write.tucson by Andy Bunn

See Also

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

Examples

```
library(utils)
data(co021)
fname <- write.compact(rwl.df = co021,
                       fname = tempfile(fileext=".rwl"),
                       append = FALSE, prec = 0.001)
print(fname) # tempfile used for output

unlink(fname) # remove the file
```

---

write.crn *Write Tucson Format Chronology File*

Description

This function writes a chronology to a Tucson (decadal) format file.

Usage

```
write.crn(crn, fname, header = NULL, append = FALSE)
```

Arguments

crn	a data.frame containing a tree-ring chronology with two columns of the type produced by chron. The first column contains the mean value chronology, the second column gives the sample depth. Years for the chronology are determined from the row names. The chronology ID is determined from the first column name.
fname	a character vector giving the file name of the crn file.
header	a list giving information for the header of the file. If NULL then no header information will be written.
append	logical flag indicating whether to append this chronology to an existing file.

Details

This writes a standard crn file as defined according to the standards of the ITRDB at https://www.ncei.noaa.gov/pub/data/paleo/treering/treeinfo.txt. This is the decadal or Tucson format. It is an ASCII file and machine readable by the standard dendrochronology programs. Header information for the chronology can be written according to the International Tree Ring Data Bank (ITRDB) standard. The header standard is not very reliable however and should be thought of as experimental here. Do not try to write headers using dplR to submit to the ITRDB. When submitting to the ITRDB, you can enter the metadata via their website. If you insist however, the header information is given as a `list` and must be formatted with the following:

Description	*Name*	*Class*	*Max Width*
Site ID	*site.id*	`character`	6
Site Name	*site.name*	`character`	52
Species Code	*spp.code*	`character`	4
State or Country	*state.country*	`character`	13
Species	*spp*	`character`	18
Elevation	*elev*	`character`	5
Latitude	*lat*	`character` or `numeric`	5
Longitude	*long*	`character` or `numeric`	5
First Year	*first.yr*	`character` or `numeric`	4
Last Year	*last.yr*	`character` or `numeric`	4
Lead Investigator	*lead.invs*	`character`	63
Completion Date	*comp.date*	`character`	8

See examples for a correctly formatted header list. If the width of the fields is less than the max width, then the fields will be padded to the right length when written. Note that *lat* and *long* are really *lat*`*100` or *long*`*100` and given as integral values. E.g., 37 degrees 30 minutes would be given as 3750.

This function takes a single chronology with sample depth as input. This means that it will fail if given output from `chron` where *prewhiten* == `TRUE`. However, more than one chronology can be appended to the bottom of an existing file (e.g., standard and residual) with a second call to `write.crn`. However, the ITRDB recommends saving and publishing only one chronology per file. The examples section shows how to circumvent this. The output from this function might be suitable for publication on the ITRDB although the header writing is clunky (see above) and rwl files are much better than crn files in terms of usefulness on the ITRDB.

Value

fname

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

`chron`, `read.crn`

Examples

```
library(utils)
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi)
fname1 <- write.crn(ca533.crn, tempfile(fileext=".crn"))
print(fname1) # tempfile used for output

## Put the standard and residual chronologies in a single file
## with ITRDB header info on top. Not recommended.
ca533.crn <- chron(ca533.rwi, prewhiten = TRUE)
ca533.hdr <- list(site.id = "CAM", site.name = "Campito Mountain",
                  spp.code = "PILO", state.country = "California",
                  spp = "Bristlecone Pine", elev = "3400M", lat = 3730,
                  long = -11813, first.yr = 626, last.yr = 1983,
                  lead.invs = "Donald A. Graybill, V.C. LaMarche, Jr.",
                  comp.date = "Nov1983")
fname2 <- write.crn(ca533.crn[, -2], tempfile(fileext=".crn"),
                    header = ca533.hdr)
write.crn(ca533.crn[, -1], fname2, append = TRUE)
print(fname2) # tempfile used for output

unlink(c(fname1, fname2)) # remove the files
```

`write.rwl` *Write Ring Width File*

Description

This function writes a `rwl` object to a file in one of the available formats.

Usage

```
write.rwl(rwl.df, fname, format = c("tucson", "compact", "tridas", "sheet", "csv"), ...)
```

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"`, `"tridas"` or `"sheet"`. Tucson format is the default. `"sheet"` is handled by `write.sheet`, and `"csv"` is accepted as a synonym for it.
`...`	arguments specific to the function implementing the operation for the chosen format.

Details

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

Value

fname

Author(s)

Mikko Korpela

See Also

write.crn, write.tucson, write.sheet, write.compact, write.tridas, read.rwl

Examples

```
library(utils)
data(co021)
co021.hdr <- list(site.id = "CO021",
                  site.name = "SCHULMAN OLD TREE NO. 1, MESA VERDE",
                  spp.code = "PSME", state.country = "COLORADO",
                  spp = "DOUGLAS FIR", elev = 2103, lat = 3712,
                  long = -10830, first.yr = 1400, last.yr = 1963,
                  lead.invs = "E. SCHULMAN", comp.date = "")
fname <- write.rwl(rwl.df = co021, fname = tempfile(fileext=".rwl"),
                   format = "tucson", header = co021.hdr,
                   append = FALSE, prec = 0.001)
print(fname) # tempfile used for output

unlink(fname) # remove the file
```

---

write.sheet *Write Ring Widths to a Spreadsheet File*

Description

Write a ring width object to a file in "spreadsheet" layout: years down the rows, series across the columns, and the years in the first column.

Usage

```
write.sheet(rwl.df, fname, sep = ",", dec = ".",
            layout = c("wide", "long"), prec = NULL, na.string = "",
            year.name = "Year")
```

Arguments

rwl.df	a data.frame of ring widths with the series in columns and the years as row names, as returned by read.rwl.
fname	a character vector giving the file name to write.
sep	the field separator, a single character.
dec	the decimal mark, "." or ",". May not be the same character as sep.
layout	the shape to write. "wide", the default, is years down the rows and series across the columns. "long" is a "long" or "tidy" file of series, year, value triples. Note that this is unrelated to the long argument of read.crn and read.tucson, which refers to a wider fixed-width year field.
prec	the precision to write, in mm, or NULL to take it from the object. Unlike write.tucson this is not restricted to 0.01 and 0.001.
na.string	character string written for a missing value. The default is an empty cell.
year.name	character string used as the header of the first column.

Details

The file is written in the layout that read.sheet reads: a header row of the year column's name followed by the series IDs, then one row per year.

Precision:

prec controls how many decimal places are written. Left at NULL it is taken from the "dplR.provenance" attribute that read.sheet and read.tucson attach to the objects they return, and failing that it is inferred from the data. This is what keeps a width read at 0.001 mm from being written back as 0.5670000000000001.

A precision that was derived rather than given is checked against the data before it is used, and if rounding there would discard real digits the file is written at full precision with a warning instead. A precision passed explicitly is an instruction and is obeyed, so prec = 0.1 rounds.

Trailing zeros are kept: at prec = 0.001 a width of 0.51 is written as 0.510.

Long format:

With layout = "long" the file holds one row per observation — the series ID, the year and the value — with missing values left out. Rows are written one series at a time, years ascending within a series. year.name names the year column here too.

A year in which *every* series is NA has no rows. An **interior** such year still survives a round trip, because read.sheet rebuilds the span from the earliest and latest year in the file and the year reappears as a row of NA in its original place. A **leading or trailing** one has nothing outside it to pin the span and is lost; write.sheet warns when it writes a file in that state. Write with layout = "wide" to keep those years.

A ring width of zero is a locally absent ring and is written as a zero. A missing value is written as na.string. The two are different statements about a tree in a year and are never written the same way.

Series IDs are written verbatim. An ID containing the separator, a double quote or a line break is quoted using the usual comma-separated-value escape.

Value

The function returns `fname`, the name of the file written.

Author(s)

Andy Bunn

See Also

`read.sheet`, `write.rwl`, `write.tucson`

Examples

```
library(utils)
data(ca533)
tmpName <- tempfile(fileext = ".csv")

# write and read back
write.sheet(ca533, tmpName)
bar <- read.sheet(tmpName, verbose = FALSE)

# the data survives the round trip unchanged. Provenance describes the file an
# object was read from, so it is dropped from both sides before comparing.
no.prov <- function(x) { attr(x, "dplR.provenance") <- NULL; x }
identical(no.prov(ca533), no.prov(bar))

unlink(tmpName)
```

---

write.tridas *Write Tree Ring Data Standard (TRiDaS) file*

Description

This function writes measured or derived (standardized, averaged) series of values to a TRiDaS format file. Some metadata are also supported.

Usage

```
write.tridas(rwl.df = NULL, fname, crn = NULL, prec = NULL, ids = NULL,
    titles = NULL, crn.types = NULL, crn.titles = NULL,
    crn.units = NULL, tridas.measuring.method = NA,
    other.measuring.method = "unknown", sample.type = "core",
    wood.completeness = NULL, taxon = "",
    tridas.variable = "ring width", other.variable = NA,
    project.info = list(type = c("unknown"), description = NULL,
                        title = "", category = "", investigator = "",
                        period = ""),
    lab.info = data.frame(name = "", acronym = NA, identifier = NA,
```

```
                        domain = "", addressLine1 = NA,
                        addressLine2 = NA, cityOrTown = NA,
                        stateProvinceRegion = NA, postalCode = NA,
                        country = NA),
    research.info = data.frame(identifier = NULL, domain = NULL,
                               description = NULL),
    site.info = list(type = "unknown", description = NULL, title = ""),
    random.identifiers = FALSE, identifier.domain = lab.info$name[1],
    ...)
```

Arguments

rwl.df	data.frame containing tree-ring ring widths in millimetres with the series in columns and the years as rows. The series IDs are the column names and the years are the row names. This type of data.frame is produced by read.tucson, read.compact and read.tridas. Defaults to NULL: no measurement series are written.
fname	character vector giving the file name of the rwl file
crn	data.frame or a list of data.frames containing tree-ring chronologies. Accepts data.frames of the type produced by chron. Additionally, allows several chronologies per data.frame. Any column of the data.frame(s) with a name starting with "samp.depth" is interpreted as a sample depth. The rest of the columns are interpreted as chronologies whose titles are determined from the column names (optionally from parameter *crn.titles*). Chronology columns and sample depth columns are paired in order so that the first chronology gets the first sample depth column, second chronology gets the second set of sample depths, etc. If there are less sample depth columns than chronologies, the sample depths are recycled. Defaults to NULL: no chronologies are written.
prec	optional numeric indicating the rounding precision of the output file when writing the data contained in *rwl.df*. Defaults to NULL: no rounding is done and the measurements are written in (non-integer) millimetres. Possible numeric values are 0.001, 0.01, 0.05, 0.1, 1, 10, 100 and 1000, which cause the data to be transformed to micrometres, 1/100th millimetres, 1/20th millimetres, 1/10 millimetres, (millimetres), centimetres, decimetres or metres, respectively, and then rounded to integral values. Data rounded to decimetres are written in centimetres (values always ending in zero). Otherwise, the matching unit is used in the file.
ids	optional data.frame with column one named "tree" giving the numeric ID of the tree, column two named "core" giving the numeric ID of the core, optional column three named "radius" giving the numeric ID of the radius, and optional column four named "measurement" giving the numeric ID of the measurement. If column "measurement" exists, column "radius" must also exist. Defaults to one core, radius and measurement per tree:

```
data.frame(tree=1:n.col, core=rep(1,n.col),
           radius=rep(1,n.col), measurement=rep(1,n.col))
```

where *n.col* is the number of columns in *rwl.df*.

`titles` optional `data.frame` with column one named `"tree"` giving the title of the tree, column two named `"core"` giving the title of the core, column three named `"radius"` giving the title of the radius, and column four named `"measurement"` giving the title of the measurement. By default, *titles* is `NULL`, and the title hierarchy is automatically created out of the column names of *rwl.df*, taking *ids* into account.

`crn.types` `character` vector or a `list` of `character` vectors giving the types of the derived series in *crn*. A single vector is interpreted as one type per `data.frame` in *crn*, recycled if not long enough. A list of vectors is interpreted as one list element per `data.frame`. In this case, the list is recycled to the correct length. After that, the vectors inside the list are recycled to match the number of derived series in each `data.frame` of *crn*. The default is to write empty '`<type>`' elements.

`crn.titles` optional `character` vector or a `list` of `character` vectors giving the titles of the derived series in *crn*. The interpretation is the same as with *crn.types*, except that the default is to derive the titles from the column names of *crn*. Also `NA` means that the column name is used.

`crn.units` optional `character` vector or a `list` of `character` vectors giving the units of the derived series in *crn*. The interpretation is the same as with *crn.types*, except that the default is to mark the series as '`<unitless>`'. Also `NA` means '`<unitless>`'.

`tridas.measuring.method`
`character` vector giving the measuring method used to acquire each series of *rwl.df*. Partial matching is used to replace these with the complete terms in `tridas.vocabulary`. If the vector is shorter than the number of columns in *rwl.df*, it is recycled to the correct length. The default is to use the information in *other.measuring.method* instead. Also, `NA` in any position of the vector means that the measuring method information for that series is looked up in *other.measuring.method*.

`other.measuring.method`
`character` vector giving the measuring method used to acquire each series of *rwl.df*. In contrast to *tridas.measuring.method*, these are free-form strings in English. If the vector is shorter than the number of columns in *rwl.df*, it is recycled to the correct length. The default value is `"unknown"`.

`sample.type` optional `character` vector giving the type of the samples, corresponding to `"core"` in *ids*. The length of the vector, however, must match the number of columns in *rwl.df*, or it is recycled to the correct length. If there are several measurements per sample, some elements of *sample.type* are redundant. The default is to use `"core"` for all series.

`wood.completeness`
optional `data.frame` giving wood completeness information for the measurement series in *rwl.df*. The number of rows must match the number of columns in *rwl.df*. The columns are expected to be a subset of the following (descriptions almost directly quoted from TRiDaS specification):

n.unmeasured.inner Field for recording whether there are any rings at the inner (i.e. towards pith) edge of the sample that have not been measured. Typically used to note when rings are too damaged to measure. Non-negative integral value.

n.unmeasured.outer Field for recording whether there are any rings at the outer (i.e. towards bark) edge of the sample that have not been measured. Typically used to note when rings are too damaged to measure. Non-negative integral value.

pith.presence Whether the pith is present or absent. Each element must be a partial match with the contents of category `"complex presence / absence"` in `tridas.vocabulary`.

heartwood.presence Whether the outer (youngest) heartwood is present and if so whether it is complete.
Category `"complex presence / absence"` in `tridas.vocabulary`.

n.missing.heartwood Estimated number of missing heartwood rings to the pith. Non-negative integral value.

missing.heartwood.foundation Description of the way the estimation of how many heartwood rings are missing was made and what the certainty is. Free-form string.

sapwood.presence Whether the sapwood is present or not.
Category `"complex presence / absence"`.

n.sapwood Number of sapwood rings measured. Non-negative integral value.

last.ring.presence Last ring under the bark is present or absent.
Category `"presence / absence"`.

last.ring.details If the last ring under the bark is present, include information about the completeness of this ring and/or season of felling. Free-form string.

n.missing.sapwood Estimated number of missing sapwood rings to the bark. Non-negative integral value.

missing.sapwood.foundation Description of the way the estimation of how many sapwood rings are missing was made and what the certainty is. Free-form string.

bark.presence Bark is present or absent.
Category `"presence / absence"` in `tridas.vocabulary`.

`taxon` `character` string. The most detailed taxonomic name known for this element (species, genus, family etc). Preferably from the Catalogue of Life controlled vocabulary. The same string is used for all of *rwl.df*. The default value is an empty string, but a proper value should really be given.

`tridas.variable`
`character` string. Measured variable (ring width, earlywood, latewood etc) taken from the TRiDaS controlled vocabulary (`tridas.vocabulary`, category `"variable"`). The same string is used for all of *rwl.df*. Defaults to `"ring width"`.

`other.variable` `character` string. Measured variable as a free-form string. The same string is used for all of *rwl.df*. This is only used if *tridas.variable* is `NA`.

`project.info` `list` containing information about the project. Elements are the following (includes quotes from the TRiDaS specification):

type `character` vector. The type(s) of the project. Defaults to `"unknown"`.

description `character` string. A description of the project. Defaults to `NULL`: no description.

title `character` string. The title of the project. Defaults to an empty string.

category `character` string. Category of research this project falls into. Defaults to an empty string.

investigator `character` string. Principal investigator of this project. Defaults to an empty string.

period `character` string. When the dendrochronological project took place. Could consist of a start- and end-date. If unknown it should be estimated. Defaults to an empty string.

`lab.info` `data.frame`. Information about the dendrochronological research laboratories where this work was done. One row per laboratory. Defaults to one laboratory with an empty name and no other information. The columns are expected to be a subset of the following:

name Name of the laboratory

acronym Optional acronym of the laboratory

identifier Optional identifier of the laboratory

domain The domain which the identifier of the laboratory is applicable to. Could be the URL of the organization's server or the name of the organization as long as it is not ambiguous.

addressLine1 First address line

addressLine2 Second address line

cityOrTown City or town

stateProvinceRegion State, province or region

postalCode Postal code. Beware of ignored leading zeros if these are given in `numeric` or `integer` type values. It is always safe to use `character` values.

country Country

`research.info` optional `data.frame` with information about the systems in which the research project is registered. Columns are the following:

identifier Identifier

domain Domain which the identifier is applicable to

description General description

`site.info` `list` containing information about the site ('<object>'). Elements are the following, and all are `character` strings:

type Type of the site. Defaults to "unknown".

description Description. Defaults to no description.

title Title of the site. Defaults to an empty string.

`random.identifiers`

`logical` flag. If `TRUE`, unique random identifiers are created with `uuid.gen` and attached to each '<project>' (one in the file), 'object' (site, one in the file), '<element>' (tree), '<sample>' (core), '<radius>', '<measurementSeries>' (measurement) and '<derivedSeries>' element in the resulting TRiDaS file.

`identifier.domain`

`character` string. The domain which the random identifiers are applicable to. Could be the URL of the organization's server or the name of the organization as long as it is not ambiguous. Defaults to the name of the first laboratory in *lab.info*.

`...` Unknown arguments are accepted but not used.

Details

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

Value

fname

Note

This is an early version of the function. Bugs are likely to exist, and parameters are subject to change.

Author(s)

Mikko Korpela

References

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

See Also

write.rwl, write.tucson, write.compact, write.crn, read.tridas

Examples

```
library(utils)
## Not run:
## Write raw ring widths
data(co021)
fname1 <- write.tridas(rwl.df = co021,
    fname = tempfile(fileext=".xml"), prec = 0.01,
    site.info = list(title = "Schulman old tree no. 1, Mesa Verde",
                     type = "unknown"),
    taxon = "Pseudotsuga menziesii var. menziesii (Mirb.) Franco",
    project.info = list(investigator = "E. Schulman",
                        title = "", category = "",
                        period = "", type = "unknown"))
print(fname1) # tempfile used for output

## Write mean value chronology of detrended ring widths
data(ca533)
ca533.rwi <- detrend(rwl = ca533, method = "ModNegExp")
ca533.crn <- chron(ca533.rwi, prewhiten = TRUE)
fname2 <- write.tridas(crn = ca533.crn,
    fname = tempfile(fileext=".xml"),
    taxon = "Pinus longaeva D.K. Bailey",
    project.info =
        list(investigator = "Donald A. Graybill, V.C. LaMarche, Jr.",
             title = "Campito Mountain", category = "",
```

```
             period = "", type = "unknown"))
print(fname2) # tempfile used for output

unlink(c(fname1, fname2)) # remove the files

## End(Not run)
```

---

write.tucson *Write Tucson Format Chronology File*

Description

This function writes a chronology to a Tucson (decadal) format file.

Usage

```
write.tucson(rwl.df, fname, header = NULL, append = FALSE,
             prec = 0.01, mapping.fname = "", mapping.append = FALSE,
             long.names = FALSE, fill.internal.NA = NULL,
             extra.chars = c("-", "_", "."), ...)
```

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.
fill.internal.NA	what to do with interior NA, i.e. years between a series' first and last measurement where *rwl.df* holds no value. The default, NULL, invents nothing and writes them as the missing-data sentinel -999 (see Details). Anything else is passed

to `fill.internal.NA` before the file is written, so *fill.internal.NA* = `0` fills the gaps with zero and `"Mean"`, `"Spline"` or `"Linear"` interpolate. This is the same argument, with the same meaning and default, as in `read.tucson`.

`extra.chars` a `character` vector of characters to allow in series IDs in addition to `a`–`z`, `A`–`Z` and `0`–`9`. The default allows the hyphen, the underscore and the period, all of which occur routinely in ITRDB IDs. Given either as single characters or as one string, so `c("-", "_")` and `"-_"` are the same request. Use *extra.chars* = `character(0)` to allow nothing beyond alphanumerics, which is what **dplR** did before version 1.8.0. Whitespace, control characters and `"#"` are refused: the first two break the fixed-width layout, and `"#"` is the default *comment.char* of `read.tucson`, which discards any line containing it.

`...` Unknown arguments are accepted but not used.

Details

This writes a standard rwl file as defined according to the standards of the ITRDB at https://www.ncei.noaa.gov/pub/data/paleo/treering/treeinfo.txt. This is the decadal or Tucson format. It is an ASCII file and machine readable by the standard dendrochronology programs. Header information for the rwl can be written according to the International Tree Ring Data Bank (ITRDB) standard. The header standard is not very reliable however and should be thought of as experimental here. Do not try to write headers using dplR to submit to the ITRDB. When submitting to the ITRDB, you can enter the metadata via their website. If you insist however, the header information is given as a `list` and must be formatted with the following:

Description	*Name*	*Class*	*Max Width*
Site ID	*site.id*	`character`	5
Site Name	*site.name*	`character`	52
Species Code	*spp.code*	`character`	4
State or Country	*state.country*	`character`	13
Species	*spp*	`character`	18
Elevation	*elev*	`character`	5
Latitude	*lat*	`character` or `numeric`	5
Longitude	*long*	`character` or `numeric`	5
First Year	*first.yr*	`character` or `numeric`	4
Last Year	*last.yr*	`character` or `numeric`	4
Lead Investigator	*lead.invs*	`character`	63
Completion Date	*comp.date*	`character`	8

See examples for a correctly formatted header list. If the width of the fields is less than the max width, then the fields will be padded to the right length when written. Note that *lat* and *long* are really *lat* `*` `100` or *long* `*` `100` and given as integral values. E.g., 37 degrees 30 minutes would be given as 3750.

Series can be appended to the bottom of an existing file with a second call to `write.tucson`. The output from this file is suitable for publication on the ITRDB.

The function is capable of altering excessively long and/or duplicate series IDs to fit the Tucson specification. Additionally, characters outside `a`–`z`, `A`–`Z`, `0`–`9` and *extra.chars* will be removed. If

series IDs are changed, a single warning is shown giving the reasons and the first few renamings as `old -> new`. The user may also wish to print the full list (see *mapping.fname* in Arguments).

Renaming is worth attention because it is not confined to the series that prompted it. Removing a character shortens a name, and a shortened name can collide with one that needed no change; the duplicate pass then renames *both*. Before **dplR** 1.8.0, a file holding `"CC1-1"` and `"CC11"` was written out as `"CC110"` and `"CC111"`, neither of which is an ID in the input.

The Tucson format does not restrict the character set of a series ID: the ITRDB description names only the columns the ID occupies. Hyphens, underscores and periods are common in real IDs, they do not disturb the fixed-width layout, and `read.tucson` reads them back unchanged, so they are written as they stand rather than removed. There is one position where a hyphen cannot go: column 8, which is where a minus sign distinguishes a 5-character year before -999 from an 8-character series ID. An 8-character ID ending in a hyphen is therefore shortened by that hyphen, with a warning. This can only arise when *long.names* = `TRUE`; at the default width an ID stops at column 6.

Interior `NA` – years between a series' first and last measurement where *rwl.df* records nothing – are written as `-999`, the negative value the Tucson format uses to mark missing data, at both precisions. `read.tucson` reads any negative value that is not the stop marker back as `NA`, so a `read.tucson`–`write.tucson`–`read.tucson` cycle preserves the gaps. Note that `read.tucson.legacy`, and other dendrochronology programs, will read those cells as a ring width of zero. A message is given whenever gaps are written.

Before **dplR** 1.8.0 interior `NA` were written as `-999` at *prec* = `0.01` but as `0` at *prec* = `0.001`. A zero ring width is a locally absent ring, which is a real observation about a tree in a year, and it is not the same statement as "not measured"; the two are no longer conflated. Use *fill.internal.NA* = `0` to get the old *prec* = `0.001` output. Files with no interior `NA` are written exactly as before.

A series that is entirely `NA` has nothing to write. It is skipped with a warning naming the series, and the rest of the file is written normally.

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`, `read.tucson.legacy`, `fill.internal.NA`, `write.rwl`, `write.compact`, `write.tridas`

Examples

```
library(utils)
data(co021)
```

```
co021.hdr <- list(site.id = "CO021",
                  site.name = "SCHULMAN OLD TREE NO. 1, MESA VERDE",
                  spp.code = "PSME", state.country = "COLORADO",
                  spp = "DOUGLAS FIR", elev = "2103M", lat = 3712,
                  long = -10830, first.yr = 1400, last.yr = 1963,
                  lead.invs = "E. SCHULMAN", comp.date = "")
fname <- write.tucson(rwl.df = co021, fname = tempfile(fileext=".rwl"),
                      header = co021.hdr, append = FALSE, prec = 0.001)
print(fname) # tempfile used for output

unlink(fname) # remove the file

## Interior NA survive a write and a read
gappy <- data.frame(SER01 = round(seq(0.5, 2, length.out = 40), 3),
                    row.names = 1901:1940)
gappy[15:19, 1] <- NA  # five years with no measurement
fname2 <- write.tucson(gappy, tempfile(fileext=".rwl"), prec = 0.001)
back <- read.tucson(fname2, verbose = FALSE)
identical(is.na(back$SER01), is.na(gappy$SER01)) # TRUE

## The old behaviour, if you want it: gaps filled with zero
fname3 <- write.tucson(gappy, tempfile(fileext=".rwl"), prec = 0.001,
                       fill.internal.NA = 0)
read.tucson(fname3, verbose = FALSE)$SER01[15:19] # zeros, not NA

unlink(c(fname2, fname3))

## Hyphens and underscores in series IDs survive a round trip
dashed <- data.frame(`CC1-1` = 1:5, CC11 = 6:10, `CC_5` = 11:15,
                     `CC.7` = 16:20,
                     check.names = FALSE, row.names = 1901:1905)
fname4 <- write.tucson(dashed, tempfile(fileext=".rwl"))
names(read.tucson(fname4, verbose = FALSE))

## The pre-1.8.0 rule, if you want it: note that stripping the hyphen from
## CC1-1 collides with CC11, so both series are renamed
fname5 <- write.tucson(dashed, tempfile(fileext=".rwl"),
                       extra.chars = character(0))
names(read.tucson(fname5, verbose = FALSE))

unlink(c(fname4, fname5))
```

`xdate.floater` *Crossdate an undated series*

Description

Pulls an undated (or misdated) series through a dated `rwl` object in order to establish possible dates for the series.

Usage

```
xdate.floater(rwl, series, series.name = "Unknown", min.overlap = 50,
              n = NULL, nyrs = NULL, prewhiten = TRUE,
              ar.order.max = NULL, biweight = TRUE,
              method = c("spearman", "pearson", "kendall"),
              make.plot = TRUE, return.rwl = TRUE, verbose = TRUE)

## S3 method for class 'floater'
print(x, ...)

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

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric vector of ring widths, e.g. a single column from an rwl object.
series.name	a character string giving a name for the undated series. Defaults to "Unknown".
min.overlap	a positive integer giving the minimum number of years of overlap required between the series and the master chronology at each search position. Defaults to 50.
n	NULL or a positive integer giving the filter length for the hanning filter used for removal of low frequency variation prior to correlation. NULL skips the filter.
nyrs	NULL or a number greater than zero. If not NULL, each series is divided by a smoothing spline with this rigidity (see caps) to remove low-frequency variation, as detrend does with method = "Spline". A value of 1 or less is taken as a proportion of each series' length. Unlike the hanning filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
prewhiten	logical flag. If TRUE (the default) each series is whitened using ar prior to correlation.
ar.order.max	NULL or a positive integer giving the maximum order of the ar model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If NULL, the order is chosen by AIC up to the ar default, which on long series can be 20 or more. Requires *prewhiten* = TRUE.
biweight	logical flag. If TRUE (the default) a robust mean is used when building the master chronology via tbrm.
method	the correlation coefficient to use. One of "spearman" (default), "pearson", or "kendall". See cor.test.
make.plot	logical flag. If TRUE (the default) a two-panel plot is produced showing the segments and the correlation surface. See Details.
return.rwl	logical flag. If TRUE (the default) a "floater" list object is returned. If FALSE only the correlation statistics data.frame is returned.
verbose	logical flag. If TRUE (the default) a brief summary is printed to the console.
x	a "floater" object produced by xdate.floater.
...	additional arguments — currently ignored.

Details

The undated series is slid along the master chronology built from `rwl` (using the leave-one-out principle) and the correlation between the series and the master is computed at each position with sufficient overlap. The position with the highest correlation gives the proposed dates.

Both series and master are optionally prewhitened and/or smoothed with a Hanning filter before correlation, consistent with the approach used in `corr.series.seg`.

The two-panel plot (produced when `make.plot = TRUE`, or by calling `plot()` on the returned object) shows: (1) a segment plot of the master series with the floater at its best-fit position highlighted in green; and (2) the correlation at each end-year searched, overlaid with a dashed significance line and a light-blue polygon showing the 5th–95th percentile band of the typical interseries correlation in the master chronology. The dark-blue horizontal line shows the median interseries correlation. The best-fit position is marked with green points and a dashed segment.

Value

If `return.rwl = TRUE` (the default), an object of class `"floater"` is returned. This is a named list with the following elements:

`series.name`	the name given to the undated series.
`floaterCorStats`	a `data.frame` with columns `first`, `last`, `r`, `p`, and `n` giving the start year, end year, correlation, p-value, and sample size at each search position.
`rwlCombined`	an `rwl` object containing the dated master series combined with the undated series placed at its best-fit dates.
`rwlOut`	an `rwl` object containing only the undated series at its best-fit dates.

If `return.rwl = FALSE`, only the `floaterCorStats` `data.frame` is returned.

Note

This function is experimental and may change in future releases. Users should always verify proposed dates visually and against the physical wood.

The choice of `min.overlap` affects both which search positions are evaluated and the reliability of the correlations at those positions. A short `min.overlap` allows the series to be placed near the edges of the master chronology where little overlap exists, producing correlations based on few observations that may be spuriously high. A very long `min.overlap` restricts the search to positions where the series is well inside the master, which is statistically conservative but can cause the function to miss the correct dates entirely if the true position lies near an edge, or to produce wrong dates if the best correlation within the constrained search window happens to be a false match. As a rule, `min.overlap` should be long enough to produce a stable correlation (50 years is a common starting point for annual tree-ring data) but no longer than roughly half the length of the undated series. The dashed significance line in the plot is calculated from the overlap length at each position and provides a useful guide: positions with short overlaps will have high significance thresholds and should be interpreted with caution.

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

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

Examples

```
library(utils)
data(co021)
summary(co021)

# Remove a series and try to recover its dates
foo <- co021[, "645232"]
bar <- co021
bar$"645232" <- NULL
out <- xdate.floater(bar, foo, min.overlap = 50, series.name = "645232")
print(out)

# A longer series. With min.overlap = 100 the correct dates are recovered.
# With min.overlap = 200 the search window is so constrained that the true
# position is excluded and the function returns the best match within the
# remaining positions, which is a false fit. Compare the two results.
foo <- co021[, "646118"]
bar <- co021
bar$"646118" <- NULL
out <- xdate.floater(bar, foo, min.overlap = 100, series.name = "646118")
out <- xdate.floater(bar, foo, min.overlap = 200, series.name = "646118")
```

---

xdate.report *COFECHA-Style Crossdating Report*

Description

Builds a crossdating report in the layout of the COFECHA output carried by the International Tree-Ring Data Bank (ITRDB) correlation-stats files: a summary header, the correlation of each series by segment with 'A' and 'B' flags, descriptive statistics for each series, and the findings of rwl.check, with a record of how the report was made.

Usage

```
xdate.report(x, seg.length = 50, bin.floor = 100, nyrs = 32,
             prewhiten = TRUE, ar.order.max = 3,
             pcrit = 0.01, lag.max = 10,
             method = c("pearson", "spearman", "kendall"),
             biweight = TRUE, check = TRUE, meta = list(),
             title = NULL)

## S3 method for class 'xdate.report'
format(x, type = c("text", "markdown", "html"),
       bins.per.page = 20, ...)
```

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

write.xdate.report(x, fname, type = NULL, ...)
```

Arguments

x	an rwl object, or the path to a file that read.rwl can read. For format, print and write.xdate.report, an object of class "xdate.report".
seg.length	an even integral value giving the length of the segments in years. Reduced, with a note in the report, when the record is too short for it.
bin.floor	a non-negative integral value giving the base for locating the first segment, as in corr.rwl.seg. Stepped down to 50, 10 and then 0, with a note, if no segment can be tested at the value given.
nyrs	NULL or a number greater than zero giving the rigidity of the smoothing spline each series is divided by before correlating, as in corr.rwl.seg. The default is COFECHA's 32 years.
prewhiten	logical flag. If TRUE each filtered series is prewhitened with an ar model.
ar.order.max	NULL or a positive integer giving the largest order of the ar model used to prewhiten. The default, 3, is lower than corr.rwl.seg's: after the spline, AIC chooses an order near 20, and prewhitening loses that many years at the start of every series. NULL leaves the order to AIC.
pcrit	a number between 0 and 1 giving the critical value for the one-tailed correlation test. The default, 0.01, is COFECHA's.
lag.max	the largest shift, in years, at which each segment is also correlated with the master. Reduced if it is not less than the segment length.
method	the correlation coefficient, "pearson" (the default, as in COFECHA), "spearman" or "kendall".
biweight	logical flag. If TRUE the master is a robust mean computed with tbrm.
check	logical flag. If TRUE the report includes the errors and warnings from rwl.check.
meta	a list of fields for the report header, any of site.name, site.location, species, investigators, latitude, longitude, elevation, checked.by and chronology.file. site.name and species are taken from the header of a Tucson file when they are not given.
title	the title printed on each part. Defaults to the file name without its extension, or the name of the object.
type	the layout: "text", the fixed-width layout of the ITRDB files; "markdown", headings and tables for a page that renders; or "html", a single self-contained page with plain tables, the flagged segments shaded and a small inline stylesheet. For write.xdate.report, NULL chooses Markdown for a file name ending in '.md' or '.markdown', HTML for '.html' or '.htm', and text otherwise.
fname	the name of the file to write.
bins.per.page	the number of segments printed side by side in the correlation table before it continues on a new page, as COFECHA does.

`...` further arguments to `format`: `print` and `write.xdate.report` pass `bins.per.page` on, and `print` also `type`.

Details

The segment correlations and flags come from `corr.rwl.seg` with *lag.max* set: each segment of each series is correlated with a leave-one-out master, at its dated position and at every shift up to *lag.max* years. A segment is flagged 'B' when some other position correlates better than the dated one, and 'A' when the dated position is the best tested but its correlation is under the critical value. With Pearson correlation the critical value printed in the report is exactly the one the test uses. Lags follow COFECHA: a negative lag means rings are probably missing from the series. The report lists each flagged segment with its lag and the gain in correlation, because a B flag is a hypothesis to check on the wood, and the size of the gain says how seriously to take it. A B flag whose best correlation is itself under the critical value is marked "weak at every lag": the segment does not crossdate anywhere in the window, the shift only wins among weak correlations, and it is better read as a low correlation than as a dating error. The letter stays B, as COFECHA's rule has it. See `corr.rwl.seg` for how to read the lags along a series.

The report differs from COFECHA in three ways, and says so in its notes. A segment is tested only where the series and the master cover all of it, at every lag, so series may show fewer segments than in COFECHA, which also tests partial segments at the ends of a series, and no flag can be raised at the ends of the record. The filtered block of the descriptive statistics describes the series that dplR correlated, which is not COFECHA's filtered series, and is labelled "dplR filtered". The correlation with the master is computed by dplR and usually differs from COFECHA's in the second decimal place.

A series whose spline cannot be fitted, because it has internal `NA` values or its spline is not all positive, is described but left out of the crossdating, and the report names it and says why. With fewer than three series that can be crossdated there is no master, and the report gives the descriptive statistics alone.

The averages in the summary and in the totals row are weighted by the number of years in each series, as COFECHA's are.

The report records how it was made: the dplR and R versions, the time, the MD5 checksum of the measurement file when `x` is a path, the filtering, the master, the correlation and the segments, and every setting that had to be changed to fit the data.

Value

`xdate.report` returns an object of class `"xdate.report"`, a `list` with elements

`title`, `file`, `meta`
: as given or found.

`stats`
: a `data.frame` with one row per series: the descriptive statistics, the number of segments tested and flagged, the correlation with the master, and whether the series was crossdated.

`flags`
: a `character` matrix, series by segment, of `"A"`, `"B"` or `""`, or `NULL` if nothing was crossdated.

`flagged`
: a `data.frame` with one row per flagged segment, in series then segment order, as in the report's flagged segments table, or `NULL` if nothing was crossdated. It

has no rows if nothing was flagged. Its columns are `seq` and `series` (the series' number and name), `from` and `to` (the segment's first and last years), `flag` (`"A"` or `"B"`), `r.dated` (the correlation at the dated position), `best.lag` (the lag that correlates best; negative means missing rings), `r.lag` and `gain` (the correlation at `best.lag` and its gain over `r.dated`, `NA` for an A flag), `weak` (`TRUE` for a B flag that is under the critical value even at its best lag) and `note`.

`crs`	the result of `corr.rwl.seg`, or `NULL`.
`excluded`	a `data.frame` of the series left out of the crossdating and why.
`check`	the result of `rwl.check`, or `NULL`.
`notes`	the changes made to fit the data.
`settings`	the settings asked for and used.
`provenance`	the dplR and R versions, the time, and the file and its checksum.

`format` returns the report as a `character` vector, one element per line. `print` prints it and returns `x` invisibly. `write.xdate.report` writes it to `fname` and returns `fname` invisibly.

Author(s)

Andy Bunn.

References

Holmes, R. L. (1983) Computer-assisted quality control in tree-ring dating and measurement. *Tree-Ring Bulletin*, **43**, 69–78.

See Also

`corr.rwl.seg`, `rwl.check`, `ccf.series.rwl`, `xskel.ccf.plot`

Examples

```
library(utils)
data(co021)
## The fault planted in the other crossdating examples: the 1500 ring
## deleted from series 641143
dat <- co021
x <- dat$"641143"
names(x) <- rownames(dat)
dat$"641143" <- delete.ring(x, year = 1500)
rpt <- xdate.report(dat, meta = list(site.name = "Schulman Old Tree No. 1",
                                     species = "PSME"))
rpt
## The flagged segments, as data
rpt$flagged
## Save it as fixed-width text, as Markdown, or as HTML
fn <- tempfile(fileext = ".txt")
write.xdate.report(rpt, fn)
fn.md <- tempfile(fileext = ".md")
write.xdate.report(rpt, fn.md)
```

```
head(readLines(fn.md), 12)
fn.html <- tempfile(fileext = ".html")
write.xdate.report(rpt, fn.html)
## Not run:
utils::browseURL(fn.html)

## End(Not run)
unlink(c(fn, fn.md, fn.html))
```

`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,
               nyrs = NULL, prewhiten = TRUE, ar.order.max = NULL,
               biweight = TRUE, series.x=FALSE)
```

Arguments

`rwl`	a `data.frame` with series as columns and years as rows such as that produced by `read.rwl`.
`series`	a `numeric` or `character` vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be `numeric`.
`series.yrs`	a `numeric` vector giving the years of *series*. Defaults to `as.numeric(names(`*series*`))`.
`win.start`	year to start window
`win.width`	an even integral value
`n`	`NULL` or an integral value giving the filter length for the `hanning` filter used for removal of low frequency variation.
`nyrs`	`NULL` or a number greater than zero. If not `NULL`, each series is divided by a smoothing spline with this rigidity (see `caps`) to remove low-frequency variation, as `detrend` does with `method = "Spline"`. A value of 1 or less is taken as a proportion of each series' length. Unlike the `hanning` filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
`prewhiten`	`logical` flag. If `TRUE` each series is whitened using `ar`.
`ar.order.max`	`NULL` or a positive integer giving the maximum order of the `ar` model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If `NULL`, the order is chosen by AIC up to the `ar` default, which on long series can be 20 or more. Requires *prewhiten* = `TRUE`.

biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.
series.x	logical flag indicating whether to make the series the x argument to ccf. See Details.

Details

This function produces a plot that is a mix of a skeleton plot and a cross-correlation plot. It's used in crossdating.

The top panel shows the normalized values for the master chronology (bottom half) and the series (top half) in green. The values are the detrended and standardized data (e.g., RWI).

Similarly, the black lines are a skeleton plot for the master and series with the marker years annotated for the master on the bottom axis and series on the top. The text at the top of the figure gives the correlation between the series and master (green bars) as well as the percentage of agreement between the years of skeleton bars for the series and master. I.e., if all the black lines occur in the same years the percentage would be 100%.

The bottom panels show cross correlations for the first half (left) and second half of the time series using function ccf.

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

The plot is built using the Grid package which allows for great flexibility in building complicated plots. However, these plots look best when they don't cover too wide a range of years (unless the plotting device is wider than is typical). For that reason the user will get a warning if *win.width* is greater than 100 years.

Old-school skeleton plots to print on paper are made with skel.plot.

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

ccf

Examples

```
library(utils)
data(co021)
dat <- co021
## Create a missing ring: delete the 1500 ring of series 641143 and
## drop the original from the master.  Dated from the bark, every ring
```

```
## before 1500 now sits one year late.
bad.series <- dat$"641143"
names(bad.series) <- rownames(dat)
bad.series <- delete.ring(bad.series, year = 1500)
dat$"641143" <- NULL
## after 1500 the dating is right
xskel.ccf.plot(rwl = dat, series = bad.series, win.start = 1550, win.width = 50)
## this window straddles the missing ring
xskel.ccf.plot(rwl = dat, series = bad.series, win.start = 1475, 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,
           nyrs = NULL, prewhiten = TRUE, ar.order.max = NULL,
           biweight = TRUE)
```

Arguments

rwl	a data.frame with series as columns and years as rows such as that produced by read.rwl.
series	a numeric or character vector. Usually a tree-ring series. If the length of the value is 1, the corresponding column of *rwl* is selected (by name or position) as the series and ignored when building the master chronology. Otherwise, the value must be numeric.
series.yrs	a numeric vector giving the years of *series*. Defaults to as.numeric(names(*series*)).
win.start	year to start window
win.end	year to end window
n	NULL or an integral value giving the filter length for the hanning filter used for removal of low frequency variation.
nyrs	NULL or a number greater than zero. If not NULL, each series is divided by a smoothing spline with this rigidity (see caps) to remove low-frequency variation, as detrend does with method = "Spline". A value of 1 or less is taken as a proportion of each series' length. Unlike the hanning filter, the spline removes no years from the ends of a series. Cannot be combined with *n*.
prewhiten	logical flag. If TRUE each series is whitened using ar.
ar.order.max	NULL or a positive integer giving the maximum order of the ar model used to prewhiten. Prewhitening removes as many years from the start of each series as the order of the model. If NULL, the order is chosen by AIC up to the ar default, which on long series can be 20 or more. Requires *prewhiten* = TRUE.
biweight	logical flag. If TRUE then a robust mean is calculated using tbrm.

Details

This function produces a plot that is a mix of a skeleton plot and a cross-correlation plot. It's used in crossdating.

The top panel shows the normalized values for the master chronology (bottom half) and the series (top half) in green. The values are the detrended and standardized data (e.g., RWI).

Similarly, the black lines are a skeleton plot for the master and series with the marker years annotated for the master on the bottom axis and series on the top. The text at the top of the figure gives the correlation between the series and master (green bars) as well as the percentage of agreement between the years of skeleton bars for the series and master. I.e., if all the black lines occur in the same years the percentage would be 100%.

The bottom panels show cross correlations for the first half (left) and second half of the time series using function `ccf` as `ccf(x=series,y=master,lag.max=5)`.

The plot is built using the Grid package which allows for great flexibility in building complicated plots. However, these plots look best when they don't cover too wide a range of years (unless the plotting device is wider than is typical). For that reason the user will get a warning if *win.width* is greater than 100 years.

Old-school skeleton plots to print on paper are made with `skel.plot`.

Value

None. Invoked for side effect (plot).

Author(s)

Andy Bunn. Patched and improved by Mikko Korpela.

See Also

`ccf`

Examples

```
library(utils)
data(co021)
dat <- co021
## Create a missing ring: delete the 1500 ring of series 641143 and
## drop the original from the master.  Dated from the bark, every ring
## before 1500 now sits one year late.
bad.series <- dat$"641143"
names(bad.series) <- rownames(dat)
bad.series <- delete.ring(bad.series, year = 1500)
dat$"641143" <- NULL
## after 1500 the dating is right
xskel.plot(rwl = dat, series = bad.series, win.start = 1550)
## this window straddles the missing ring
xskel.plot(rwl = dat, series = bad.series, win.start = 1475)
```

zof.anc — *Ancillary Data Corresponding to* zof.rwl

Description

This data set gives the pith offsets, distance to pith, and diameter that match the ring widths for zof.rwl – a data set of European Beech (*Fagus sylvatica*) increment cores collected at the Zofingen in Switzerland.

Usage

```
data(zof.anc)
```

Format

A data.frame containing four columns. Column one gives the series ID. Column two (PO) gives the number of rings estimated to be missing to the pith. Column three (d2pith) gives the estimated distance to the pith (mm). Column four (diam) gives the diameter at breast height (DBH) in cm.

Source

Contributed by Stefan Klesse

References

Klesse, S., Babst, F., Lienert, S., Spahni, R., Joos, F., Bouriaud, O., Carrer, M., Filippo, A.D., Poulter, B., Trotsiuk, V., Wilson, R., Frank, D.C. (2018) A combined tree ring and vegetation model assessment of European forest growth sensitivity to interannual climate variability. *Global Biogeochemical Cycles* 32, 1226 – 1240.

zof.rwl — *European Beech Ring Widths from Zofingen, Switzerland*

Description

This data set includes ring-width measurements for European Beech (*Fagus sylvatica*) increment cores collected at the Zofingen in Switzerland. There are 61 series.

Usage

```
data(zof.rwl)
```

Format

A data.frame containing 61 ring-width series in columns and 156 years in rows.

Source

Contributed by Stefan Klesse

References

Klesse, S., Babst, F., Lienert, S., Spahni, R., Joos, F., Bouriaud, O., Carrer, M., Filippo, A.D., Poulter, B., Trotsiuk, V., Wilson, R., Frank, D.C. (2018) A combined tree ring and vegetation model assessment of European forest growth sensitivity to interannual climate variability. *Global Biogeochemical Cycles* 32, 1226 – 1240.

Index

∗ **IO**
read.compact, 93
read.crn, 94
read.fh, 95
read.rwl, 100
read.sheet, 101
read.tridas, 104
read.tucson, 111
read.tucson.legacy, 114
write.compact, 163
write.crn, 164
write.rwl, 166
write.sheet, 167
write.tridas, 169
write.tucson, 175
∗ **aplot**
rasterPlot, 87
∗ **classes**
as.bai, 7
as.rwi, 9
∗ **datasets**
anos1, 7
ca533, 18
cana157, 19
co021, 34
gp.d2pith, 59
gp.dbh, 60
gp.po, 60
gp.rwl, 61
nm046, 74
wa082, 157
zof.anc, 189
zof.rwl, 189
∗ **help**
dplR-defunct, 51
∗ **hplot**
morlet, 71
plot.crn, 77
plot.crs, 78
plot.rwl, 79
seg.plot, 133
skel.plot, 139
spag.plot, 141
wavelet.plot, 158
xskel.ccf.plot, 185
xskel.plot, 187
∗ **htest**
redfit, 115
∗ **iplot**
i.detrend, 63
i.detrend.series, 64
∗ **manip**
as.bai, 7
as.rwi, 9
as.rwl, 12
bai.in, 13
bai.out, 14
ccf.series.rwl, 21
check.rwl, 24
chron, 26
chron.ars, 27
chron.ci, 29
chron.stabilized, 30
cms, 32
combine.rwl, 35
common.interval, 36
corr.rwl.seg, 37
corr.series.seg, 41
detrend, 44
detrend.series, 46
Extract.rwl, 51
fill.internal.NA, 54
insert.ring, 65
interseries.cor, 66
po.to.wc, 80
powt, 83
print.rwl.report, 86
rcs, 90

rwl.report, 128
series.rwl.plot, 136
ssf, 143
strip.rwl, 148
time.rwl, 151
treeMean, 152
wc.to.po, 160
window.rwl, 161
xdate.floater, 178
xdate.report, 181
∗ **misc**
read.ids, 96
rwi.stats.running, 121
rwl.check, 125
rwl.stats, 130
sss, 146
∗ **package**
dplR-package, 4
∗ **print**
print.redfit, 84
∗ **robust**
tbrm, 149
∗ **smooth**
ads, 5
caps, 19
pass.filt, 75
∗ **ts**
glk, 57
hanning, 62
net, 73
redfit, 115
sea, 131
∗ **univar**
gini.coef, 56
sens1, 134
sens2, 135
tbrm, 149
∗ **utilities**
latexDate, 68
latexify, 69
rasterPlot, 87
tridas.vocabulary, 153
uuid.gen, 155
.Platform, *155*
.Random.seed, *155*
[.bai *(*as.bai*)*, 7
[.data.frame, *52*
[.rwi *(*as.rwi*)*, 9
[.rwl, *8*, *10*, *36*, *45*, *113*, *162*
[.rwl *(*Extract.rwl*)*, 51
%dopar%, *45*, *124*

ads, 5, *21*, *48*, *91, 92*
anos1, 7, *127*
approx, *55*
ar, *22, 23*, *26*, *28, 29*, *38, 39*, *42*, *48*, *67*, *122*, *136*, *179*, *182*, *185*, *187*
arima, *28, 29*
as.bai, 7, *10*, *13–15*, *24, 25*
as.data.frame.rwl.check *(*rwl.check*)*, 125
as.data.frame.summary.rwi *(*as.rwi*)*, 9
as.rwi, *8*, 9, *24, 25*, *33*, *36*, *45*, *63*, *91*, *142*
as.rwl, *8*, *10, 11*, 12, *24, 25*, *103*, *125*, *144*
autoread.ids *(*read.ids*)*, 96
axis, *38*

bai *(*as.bai*)*, 7
bai.in, *7, 8*, *10*, 13, *15*, *17*
bai.out, *7, 8*, *10*, *13*, 14, *14*, *17*
bakker, 16
bigq, *116*
boot, *30*
boot.ci, *30*
butter, *76*

ca533, 18, *63*, *96*, *123*, *127*
Cairo, *87, 88*
Cairo.capabilities, *88*
call, *87*
cana157, 19
capabilities, *88*
caps, *5, 6*, 19, *22*, *38*, *42*, *47–49*, *51*, *55*, *67*, *77*, *91, 92*, *122*, *126*, *136*, *144*, *158*, *160*, *179*, *185*, *187*
captureOutput, *70*
ccf, *22, 23*, *186*, *188*
ccf.series.rwl, 21, *39, 40*, *43*, *127*, *137*, *181*, *184*
cheby1, *76*
check.rwl, 24
chron, *4*, *8*, *10, 11*, 26, *28, 29*, *32, 33*, *52, 53*, *77, 78*, *92*, *132*, *145*, *150*, *164, 165*, *170*
chron.ars, *10*, *26*, 27, *77*
chron.ci, 29
chron.stabilized, *10*, 30, *77*
cms, *8–11*, 32, *92*

co021, 34, *127*
combine.rwl, 35
common.interval, *8*, *10*, 36, *52*, *53*
compactPDF, *87*
cor, *122*, *124*
cor.test, *38*, *39*, *42*, *67*, *68*, *179*
corr.rwl.seg, *4*, *11*, *23*, 37, *52*, *67*, *78*, *79*, *123*, *124*, *126*, *127*, *137*, *182–184*
corr.series.seg, *23*, *40*, 41, *43*, *137*, *180*, *181*
crn.plot, *26*, *29*
crn.plot *(*plot.crn*)*, 77
csv *(*read.sheet*)*, 101
csv2rwl *(*read.sheet*)*, 101

data.frame, *52*
delete.ring *(*insert.ring*)*, 65
detrend, *4*, *6*, *8–11*, *22*, *24*, *26*, *27*, *29*, *30*, *33*, *38*, *39*, *42*, 44, *50*, *52*, *53*, *67*, *76*, *92*, *112*, *113*, *121–124*, *136*, *145*, *146*, *179*, *185*, *187*
detrend.series, *4*, *44*, *45*, 46, *63–65*, *143*
dev.capabilities, *88*
Devices, *45*, *140*
digest, *156*
dplR, *65*
dplR *(*dplR-package*)*, 4
dplR-defunct, 51
dplR-package, 4
dQuote, *70*

Encoding, *70*
expression, *87*
Extract.rwl, 51

ffcsaps *(*dplR-defunct*)*, 51
file, *94*, *114*
fill.internal.NA, *20*, *25*, *45*, 54, *101*, *111–113*, *176*, *177*
filter, *62*
filtfilt, *76*
foreach, *45*, *124*
format, *85*
format.xdate.report *(*xdate.report*)*, 181
formula, *49*
fread, *102*

gini.coef, 56, *131*
glk, *4*, 57, *73*, *139*
glk(), *138*
gp.d2pith, 59
gp.dbh, 60
gp.po, 60
gp.rwl, *59*, *60*, 61, *127*
Grid, *186*, *188*
grid.raster, *88*

hanning, *22*, *23*, *38*, *39*, *42*, 62, *67*, *76*, *122*, *136*, *140*, *179*, *185*, *187*
head, *45*, *52*, *53*

i.detrend, *10*, 63
i.detrend.series, 64
iconv, *95*, *102*, *111*
image, *10*
insert.ring, 65
interseries.cor, *11*, *31*, *52*, 66, *86*, *87*, *127*, *129*, *130*

latexDate, 68
latexify, 69
lines, *142*

match.call, *118*
message, *88*
morlet, 71, *158*, *160*

net, 73
nls, *47*, *48*
nm046, 74, *127*

packageVersion, *118*
par, *87*
pass.filt, 75
plot, *77*, *79*, *91*
plot.bai *(*as.bai*)*, 7
plot.crn, 77
plot.crs, *40*, 78
plot.floater *(*xdate.floater*)*, 178
plot.rwi *(*as.rwi*)*, 9
plot.rwl, *53*, 79
plotRings *(*dplR-defunct*)*, 51
png, *87*, *88*
po.to.wc, 80, *161*
pointer, *10*, 81
powt, 83, *91*
print.data.frame, *85*
print.floater *(*xdate.floater*)*, 178
print.redfit, 84, *119*

print.rwl.check *(rwl.check)*, 125
print.rwl.report, 86
print.summary.rwi *(as.rwi)*, 9
print.xdate.report *(xdate.report)*, 181

R.version, *155*
Random, *156*
rasterImage, *88*
rasterPlot, 87, *159*
rcs, *8–11*, *33*, *81*, 90, *96*, *161*
read.compact, 93, *101*, *110*, *113*, *115*, *163*, *164*, *170*, *175*
read.crn, *19*, 94, *101*, *111*, *151*, *165*, *168*
read.fh, *83*, *94*, 95, *101*, *110*, *113*, *115*, *148*
read.ids, *7*, *10*, *53*, 96, *123*, *124*, *146*, *147*, *152*
read.rwl, *4*, *7*, *10*, *13*, *15*, *18*, *22*, *24–26*, *29–32*, *34*, *36*, *37*, *42*, *44*, *52–55*, *63*, *67*, *74*, *80*, *81*, *83*, *90*, *94*, *96*, *99*, 100, *103*, *110*, *112*, *113*, *115*, *128*, *130*, *131*, *133*, *136*, *138*, *142*, *143*, *148*, *151*, *152*, *157*, *163*, *166–168*, *175*, *179*, *182*, *185*, *187*
read.sheet, *100*, 101, *101*, *111*, *168*, *169*
read.table, *103*
read.tridas, *94*, *100*, *101*, 104, *113*, *115*, *154*, *160*, *161*, *170*, *174*
read.tucson, *8*, *10*, *53*, *94*, *95*, *101–103*, *110*, 111, *114*, *115*, *125*, *127*, *129*, *157*, *168*, *170*, *176*, *177*
read.tucson.legacy, *101*, *103*, *111–113*, 114, *127*, *129*, *177*
readPNG, *88*
redfit, *4*, *85*, 115
regex, *70*
rowMeans, *152*
runcrit *(redfit)*, 115
rwi *(as.rwi)*, 9
rwi.stats, *8*, *10*, *11*, *52*, *67*, *68*, *99*, *124*, *131*, *146–149*
rwi.stats *(rwi.stats.running)*, 121
rwi.stats.legacy, *124*
rwi.stats.running, *10*, *11*, 121
rwl.check, *40*, *52*, *53*, *103*, *113*, 125, *129*, *181*, *182*, *184*
rwl.report, *8*, *10*, *86*, *87*, *126*, *127*, 128
rwl.stats, *8*, *10*, *53*, *57*, *68*, 130, *134*, *135*

save, *7*, *18*, *19*, *34*, *74*, *157*
scale, *142*
sea, 131
seg.plot, *8*, *10*, *36*, *80*, 133, *142*
sens1, *131*, 134, *135*
sens2, *131*, *134*, 135
series.rwl.plot, *23*, *39*, *40*, *43*, *67*, 136, *181*
sgc, *10*, *58*, 138
skel.plot, *23*, *40*, *43*, *82*, 139, *181*, *186*, *188*
spag.plot, *8*, *10*, *11*, *80*, *133*, 141
spline, *55*
sQuote, *70*
ssf, *10*, *77*, 143
sss, *10*, *123*, 146
strip.rwl, *10*, 148
subset.bai *(as.bai)*, 7
subset.data.frame, *52*, *53*
subset.rwi *(as.rwi)*, 9
subset.rwl, *8*, *10*
subset.rwl *(Extract.rwl)*, 51
summary, *130*
summary.bai *(as.bai)*, 7
summary.rwi, *52*
summary.rwi *(as.rwi)*, 9
summary.rwl, *86*, *87*, *129*, *130*
summary.rwl *(rwl.stats)*, 130
summary.rwl.check *(rwl.check)*, 125
supsmu, *44*, *45*, *47*, *48*
Sys.info, *155*

tail, *52*, *53*
tbrm, *22*, *23*, *26*, *27*, *29*, *31*, *38*, *39*, *42*, *67*, *91*, *136*, 149, *179*, *182*, *186*, *187*
time, *103*
time.bai *(as.bai)*, 7
time.crn *(time.rwl)*, 151
time.rwi *(as.rwi)*, 9
time.rwl, *53*, 151, *162*
time<- *(time.rwl)*, 151
time<-.bai *(as.bai)*, 7
time<-.rwi *(as.rwi)*, 9
treeMean, 152
tridas.vocabulary, 153, *171*, *172*

uuid.gen, 155, *173*

wa082, *127*, 157
wavelet.plot, *72*, *73*, *142*, 158
wc.to.po, *81*, 160
window, *8*, *162*

window.bai *(window.rwl)*, 161
window.crn *(window.rwl)*, 161
window.rwi, *10*
window.rwi *(window.rwl)*, 161
window.rwl, *8*, *52*, 161
write.compact, *94*, 163, *167*, *174*, *177*
write.crn, 164, *165*, *167*, *174*, *177*
write.rwl, *101*, *164*, 166, *169*, *174*, *177*
write.sheet, *102, 103*, *166*, 167, *167*
write.tridas, *81*, *110*, *154*, *164*, *167*, 169, *177*
write.tucson, *113*, *115*, *164*, *167–169*, *174*, 175
write.xdate.report *(xdate.report)*, 181

x11, *88*
xdate.floater, 178
xdate.report, 181
xskel.ccf.plot, *39*, *140*, *184*, 185
xskel.plot, *140*, 187

zof.anc, 189
zof.rwl, 189, *189*