Package 'burnr'

August 22, 2019

Title Forest Fire History Analysis

Version 0.5.0

Description Tools to read, write, parse, and analyze forest fire history data (e.g. FHX). Described in Malevich et al. (2018) <doi:10.1016/j.dendro.2018.02.005>.

URL https://github.com/ltrr-arizona-edu/burnr/

BugReports https://github.com/ltrr-arizona-edu/burnr/issues

Depends R (>= 3.2)

License GPL (>= 3)

Encoding UTF-8

LazyData true

Suggests testthat, knitr, rmarkdown

Imports MASS, stats, ggplot2, reshape2, plyr

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Steven Malevich [aut, cre] (<https://orcid.org/0000-0002-4752-8190>), Christopher Guiterman [ctb] (<https://orcid.org/0000-0002-9706-9332>), Ellis Margolis [ctb] (<https://orcid.org/0000-0002-0595-9005>)

Maintainer Steven Malevich <sbmalev@gmail.com>

Repository CRAN

Date/Publication 2019-08-21 22:50:02 UTC

R topics documented:

`+.fhx` *Concatenate or combine two fhx objects*

Description

Concatenate or combine two fhx objects

Usage

```
## S3 method for class 'fhx'
a + b
```

Arguments

a	An fhx object.
b	The fhx object to be append.

Value

An fhx object with the observations from a and b.

Note

Throws `stop()` if there are duplicate series names in a and b.

See Also

- `series_names()` get all the series in an fhx object.
- `get_series()` subset an fhx object to select series.
- `delete()` remove observations from an fhx object.
- `sort.fhx()` sort an fhx object.

Examples

```
data(lgr2)
data(pgm)
plot(lgr2 + pgm)
```

as.fhx *Alias to* as_fhx()

Description

Alias to as_fhx()

Usage

```
as.fhx(x)
```

Arguments

x	A data frame or list-like object to cast. Must have named elements for "year", "series", and "rec_type".

Value

x cast to an fhx object.

See Also

- fhx() constructs an fhx object.
- is_fhx() test whether object is fhx.
- make_rec_type() helpful to convert rec_type-like character vectors to full facors with proper levels.

Examples

```
data(lgr2)
example_dataframe <- as.data.frame(lgr2)
back_to_fhx <- as_fhx(example_dataframe)
```

as_fhx *Cast data frame or list-like to* fhx *object*

Description

Cast data frame or list-like to fhx object

Usage

```
as_fhx(x)
```

Arguments

x	A data frame or list-like object to cast. Must have named elements for "year", "series", and "rec_type".

Value

x cast to an fhx object.

See Also

- fhx() constructs an fhx object.
- is_fhx() test whether object is fhx.
- make_rec_type() helpful to convert rec_type-like character vectors to full facors with proper levels.

Examples

```
data(lgr2)
example_dataframe <- as.data.frame(lgr2)
back_to_fhx <- as_fhx(example_dataframe)
```

---

composite *Composite fire events in fhx object*

Description

Composite fire events in fhx object

Usage

```
composite(x, filter_prop = 0.25, filter_min_rec = 2,
  filter_min_events = 1, injury_event = FALSE, comp_name = "COMP")
```

Arguments

x	An fhx object.
filter_prop	The minimum proportion of fire events in recording series needed for fire event to be considered for composite. Default is 0.25.
filter_min_rec	The minimum number of recording series needed for a fire event to be considered for the composite. Default is 2 recording series.
filter_min_events	The minimum number of fire scars needed for a fire event to be considered for the composite. Default is 1. Fire injuries are included in this count if injury_event is TRUE.

injury_event	Boolean indicating whether injuries should be considered events. Default is FALSE.
comp_name	Character vector of the series name for the returned fhx object composite series. Default is 'COMP'.

Value

An fhx object representing the composited series. The object will be empty if there are nocomposite-worthy events.

See Also

- intervals() fire interval analysis from an fhx composite.
- sea() superposed epoch analysis.
- series_stats() basic summary stats for an fhx object.
- get_event_years() gets years for various events in an fhx object.
- count_event_position() count the number of different events in an fhx object.
- yearly_recording() count the number of "recording" events in each year of an fhx object.
- fhx() constructs an fhx object.
- as_fhx() casts data frame-like object into an fhx object.

Examples

```
data(lgr2)
composite(lgr2)

# Use with composite to get composite years:
comp <- composite(pgm, comp_name = "pgm")
event_yrs <- get_event_years(comp)["pgm"]]
print(event_yrs)
```

---

count_event_position *Count different events in an* fhx *object*

Description

Count different events in an fhx object

Usage

```
count_event_position(x, injury_event = FALSE, position, groupby)
```

Arguments

x	An fhx object.
injury_event	Optional boolean indicating whether injuries should be considered an "event". Default is FALSE.
position	Optional character vector giving the types of event positions to include in the count. Can be any combination of the following: • "unknown" • "dormant" • "early" • "middle" • "late" • "latewd" The default counts all types of event positions.
groupby	Optional named list containing character vectors that are used to count the total number of different event types. The names given to each character vector give the group's name in the output data frame.

Value

A data frame with a columns giving the event or event group and values giving the corresponding count for each event type or group.

See Also

- get_event_years() gets years for various events in an fhx object.
- yearly_recording() count the number of "recording" events in each year of an fhx object.
- series_stats() basic summary stats for an fhx object.

Examples

```
data(pgm)
count_event_position(pgm)

# As above, but considering injuries to be a type of event.
count_event_position(pgm, injury_event = TRUE)

# Count only events of a certain position, in this case, "unknown", "early",
# and "middle".
count_event_position(pgm,
  injury_event = TRUE,
  position = c("unknown", "early", "middle")
)

# Using custom "groupby" args.
grplist <- list(
  foo = c("dormant_fs", "early_fs"),
  bar = c("middle_fs", "late_fs")
```

```
)
count_event_position(pgm, groupby = grplist)
```

`count_injury` *Number of injury events in an* `fhx` *object*

Description

Number of injury events in an `fhx` object

Usage

```
count_injury(x)
```

Arguments

`x`	An `fhx` object.

Value

The number of injury events in `x`

See Also

- `count_scar()` Count the injuries in an `fhx` object.
- `series_stats()` basic statistics for series in an `fhx` object.

`count_recording` *Number of recording years in an* `fhx` *object*

Description

Number of recording years in an `fhx` object

Usage

```
count_recording(x, injury_event = FALSE)
```

Arguments

`x`	An `fhx` object.
`injury_event`	Boolean indicating whether injuries should be considered event.

Value

The number of recording events in `x`.

See Also

series_stats() basic statistics for series in an fhx object.

---

count_scar *Number of scar events in an* fhx *object*

Description

Number of scar events in an fhx object

Usage

```
count_scar(x)
```

Arguments

x	An fhx object.

Value

The number of fire scar events in x

See Also

- count_injury() Count the injuries in an fhx object.
- series_stats() basic statistics for series in an fhx object.

---

count_year_span *Number of years of an* fhx *object*

Description

Number of years of an fhx object

Usage

```
count_year_span(x)
```

Arguments

x	An fhx object.

Value

The difference between the first and last observations in the fhx object. NA will be returned if NA is in x$year.

See Also

- first_year() get first year of fhx object.
- last_year() get last year of fhx object.
- series_stats() basic statistics for series in an fhx object.

---

delete *Remove series or years from an* fhx *object*

Description

Remove series or years from an fhx object

Usage

```
delete(x, s, yr)
```

Arguments

x	An fhx object.
s	Character vector of series to remove from x.
yr	Integer vector of years to remove from x.

Details

You can combine s and yr to specify years within select series to remove.

Value

An fhx object with observations removed.

See Also

- fhx() constructs an fhx object.
- as_fhx() casts data frame-like object into an fhx object.
- series_names() get all the series in an fhx object.
- year_range() get earliest and latest year in an fhx object.
- get_year() subset an fhx object to select years.
- get_series() subset an fhx object to select series.
- get_event_years() gets years for various events in an fhx object.

Examples

```
data(lgr2)
plot(delete(lgr2, s = "LGR46"))

plot(delete(lgr2, yr = 1300:1550))
```

fhx *Constructor for* fhx *objects*

Description

Constructor for fhx objects

Usage

```
fhx(year, series, rec_type)
```

Arguments

year	An n-length numeric vector of observation years.
series	An n-length factor or character vector of observation series names.
rec_type	An n-length factor or character vector denoting the record type for each observations. Note that this needs to use a controlled vocabulary, see burnr:::rec_type_all for all possible values.

Details

Note that 'year', 'series', and 'rec_type' are pass through as.numeric(), as.factor(), and make_rec_type() the fhx object is created.

Value

An fhx object. fhx are S3 objects; specialized data frames with 3 columns:

- "year": An n-length numeric vector. The year of an observation.
- "series": An n-length factor. Giving the series name for each observation.
- "rec_type": An n-length factor with controlled vocabulary and levels. This records the type of ring or record of each observation.

See Also

- as_fhx() casts data frame-like object into fhx object.
- sort.fhx() sort an fhx object.
- is_fhx() test whether object is fhx.
- +.fhx() concatenate multiple fhx objects together.
- make_rec_type() helpful to convert rec_type-like character vectors to full facors with proper levels.
- read_fhx() Read FHX2 files.
- write_fhx() Write FHX2 files.
- plot_demograph() makes demography plots of fhx objects.

- series_stats() basic common statistical summaries of fhx objects.
- composite() create fire composites from fhx objects.
- intervals() fire interval analysis.
- sea() superposed epoch analysis.

Examples

```
x <- fhx(
  year = c(1900, 1954, 1996),
  series = rep("tree1", 3),
  rec_type = c("pith_year", "unknown_fs", "bark_year")
)
print(x)
```

---

first_year *First (earliest) year of an* fhx *object*

Description

First (earliest) year of an fhx object

Usage

```
first_year(x)
```

Arguments

x	An fhx object.

Value

The minimum or first year of series in x.

See Also

- last_year() get last year of fhx object.
- series_stats() basic statistics for series in an fhx object.

get_event_years *Get years with events for an* fhx *object*

Description

Get years with events for an fhx object

Usage

```
get_event_years(x, scar_event = TRUE, injury_event = FALSE,
  custom_grep_str = NULL)
```

Arguments

x	An fhx object.
scar_event	Boolean indicating whether years with scar events should be returned. Default is TRUE.
injury_event	Boolean indicating whether years with injury events should be returned. Default is FALSE.
custom_grep_str	Character string to pass a custom grep search pattern to search x "rec_type" column for. NULL by default.

Value

A list. Elements of the list are numeric vectors giving the years with events for each fhx series. Each element's name reflects the series' name.

See Also

- series_names() get all the series in an fhx object.
- year_range() get earliest and latest year in an fhx object.
- get_year() subset an fhx object to select years.
- get_series() subset an fhx object to select series.
- get_event_years() gets years for various events in an fhx object.
- count_event_position() count the number of different events in an fhx object.
- yearly_recording() count the number of "recording" events in each year of an fhx object.
- series_stats() basic summary stats for an fhx object.

Examples

```
data(pgm)
get_event_years(pgm, scar_event = TRUE, injury_event = TRUE)

# Passing a custom string to grep. This one identified recorder years:
get_event_years(pgm, custom_grep_str = "recorder_")

# Use with composite to get composite years:
comp <- composite(pgm, comp_name = "pgm")
event_yrs <- get_event_years(comp)[["pgm"]]
print(event_yrs)
```

---

get_series *Extract* fhx *observations for given series*

Description

Extract fhx observations for given series

Usage

```
get_series(x, s)
```

Arguments

x	An fhx object.
s	Character vector of series to extract from x.

Value

An fhx object.

See Also

- series_names() get all the series in an fhx object.
- get_year() subset an fhx object to select years
- delete() remove observations from an fhx object.

Examples

```
data(lgr2)
get_series(lgr2, "LGR46")

get_series(lgr2, c("LGR41", "LGR46"))
```

get_year *Extract* fhx *observations for given years*

Description

Extract fhx observations for given years

Usage

```
get_year(x, yr)
```

Arguments

x	An fhx object.
yr	Numeric vector of year(s) to extract from x.

Value

An fhx object.

See Also

- year_range() get earliest and latest year in an fhx object.
- get_series() subset an fhx object to select series.
- delete() remove observations from an fhx object.
- get_event_years() gets years for various events in an fhx object.

Examples

```
data(lgr2)
get_year(lgr2, 1806)

get_year(lgr2, 1805:1807)
```

inner_type *Type of observation in the first (earliest) year of an* fhx *object*

Description

Type of observation in the first (earliest) year of an fhx object

Usage

```
inner_type(x)
```

Arguments

x	An fhx object.

Value

The a factor giving the type of observation in the first observation of x.

See Also

- outer_type() get observation type in outer-most year of fhx object.
- series_stats() basic statistics for series in an fhx object.

---

intervals *Calculate fire intervals from a* composite

Description

Calculate fire intervals from a composite

Usage

```
intervals(comp, densfun = "weibull")
```

Arguments

comp	A composite instance, usually output from composite(). Should contain only one series.
densfun	String giving desired distribution to fit. Either "weibull" or "lognormal". Default is "weibull".

Value

An intervals object. intervals have components:

- "intervals" an integer vector giving the actual fire intervals.
- "fitdistr" a fitdistr object from MASS::fitdistr() representing the density function fit.
- "densfun" a string giving the name of the density function used.
- "kstest" an htest object from stats::ks.test() giving the result of a one-sample Kolmogorov-Smirnov test.
- "shapirotest" an htest object from stats::shapiro.test() giving the result of a Shapiro-Wilk normality test.
- "comp_name" a string giving the name of the interval's input composite.
- "event_range" an integer vector giving the year range (min, max) of events used to create this intervals.

See Also

- composite() to create a composite object.
- mean.intervals() gets mean fire interval.
- median.intervals() gets median fire interval.
- quantile.intervals() get fit distribution quantiles.
- plot_intervals_dist() plots intervals.
- min.intervals() gives the minimum fire interval.
- max.intervals() gives the maximum fire interval.
- print.intervals() prints common fire-interval summary statistics.

Examples

```
data(pgm)
interv <- intervals(composite(pgm))
print(interv)

mean(interv) # Mean interval

# Now fit log-normal distribution instead of Weibull.
intervals(composite(pgm), densfun = "lognormal")
## Not run:
# Boxplot of fire interval distribution.
boxplot(intervals(composite(pgm))$intervals)

## End(Not run)
```

---

is.fhx *Alias to* is_fhx()

Description

Alias to is_fhx()

Usage

```
is.fhx(x)
```

Arguments

x	An object.

Value

Boolean indicating whether x is an fhx object.

See Also

- `fhx()` constructs an `fhx` object.
- `as_fhx()` casts data frame-like object into an `fhx` object.
- `+.fhx()` concatenate multiple `fhx` objects together.

Examples

```
data(lgr2)
is_fhx(lgr2)
```

---

is.intervals *Alias to* `is_intervals()`

Description

Alias to `is_intervals()`

Usage

```
is.intervals(x)
```

Arguments

x	An R object.

Value

Boolean indicating whether x is an `intervals` object.

See Also

`intervals()` creates an `intervals` object.

is.sea *Alias to* is_sea()

Description

Alias to is_sea()

Usage

```
is.sea(x)
```

Arguments

x	An R object.

Value

Boolean indicating whether x is a sea object.

See Also

sea() creates a sea object.

is_fhx *Check if object is* fhx.

Description

Check if object is fhx.

Usage

```
is_fhx(x)
```

Arguments

x	An object.

Value

Boolean indicating whether x is an fhx object.

See Also

- fhx() constructs an fhx object.
- as_fhx() casts data frame-like object into an fhx object.
- +.fhx() concatenate multiple fhx objects together.

Examples

```
data(lgr2)
is_fhx(lgr2)
```

`is_intervals` *Check if object is fire* `intervals`

Description

Check if object is fire `intervals`

Usage

```
is_intervals(x)
```

Arguments

x	An R object.

Value

Boolean indicating whether x is an `intervals` object.

See Also

`intervals()` creates an `intervals` object.

`is_sea` *Check if object is* `sea`

Description

Check if object is `sea`

Usage

```
is_sea(x)
```

Arguments

x	An R object.

Value

Boolean indicating whether x is a `sea` object.

See Also

sea() creates a sea object.

---

last_year *Last (most recent) year of an* fhx *object*

Description

Last (most recent) year of an fhx object

Usage

```
last_year(x)
```

Arguments

x	An fhx object.

Value

The maximum or last year of series in x. NA will be returned if NA is in x$year.

See Also

- first_year() get first year of fhx object.
- series_stats() basic statistics for series in an fhx object.

---

lgr2 *Los Griegos Peak plot2 fire-history data*

Description

An fhx object with fire-history data from Los Griegos Peak, New Mexico.

Usage

```
lgr2
```

Format

An fhx object with 26 series from 1366 to 2012 CE.

See Also

lgr2_meta Los Griegos Peak metadata.

lgr2_meta *Metadata for the Los Griegos Peak fire-history dataset*

Description

A data frame with species information for the Los Griegos Peak plot2 fire-history dataset (lgr2).

Usage

```
lgr2_meta
```

Format

A `data.frame` with 26 rows and 2 variables:

- "TreeID": Name of tree series.
- "SpeciesID": Abbreviated tree species

See Also

lgr2 Log Griegos Peak fire-history data.

make_rec_type *Turn character vector into factor with proper* `fhx` *levels*

Description

Turn character vector into factor with proper `fhx` levels

Usage

```
make_rec_type(x)
```

Arguments

`x`	A character vector or factor containing one or more rec_type-like strings. This uses a controlled vocabulary, see `burnr:::rec_type_all` for list of all possible rec_type values.

Value

A factor with appropriate `fhx` levels.

See Also

- `fhx()` constructs an `fhx` object.
- `as_fhx()` casts data frame-like objects into `fhx` objects.

Examples

```
make_rec_type("null_year")

make_rec_type(c("null_year", "late_fs"))
```

`max.intervals` *Maximum interval in fire* `intervals`

Description

Maximum interval in fire `intervals`

Usage

```
## S3 method for class 'intervals'
max(x, ...)
```

Arguments

`x`	An `intervals` object.
`...`	Additional arguments passed to `max()`.

Value

Numeric or NA.

See Also

- `intervals()` to create a fire `intervals` object.
- `mean.intervals()` gets median fire interval.
- `median.intervals()` gets median fire interval.
- `quantile.intervals()` get fit distribution quantiles.
- `min.intervals()` gives the minimum fire interval.
- `print.intervals()` prints common fire-interval summary statistics.

mean.intervals *Fire* intervals *arithmetic mean*

Description

Fire intervals arithmetic mean

Usage

```
## S3 method for class 'intervals'
mean(x, ...)
```

Arguments

x	An intervals object.
...	Additional arguments passed to mean().

Value

Numeric or NA.

See Also

- intervals() to create a fire intervals object.
- median.intervals() gets median fire interval.
- quantile.intervals() get fit distribution quantiles.
- min.intervals() gives the minimum fire interval.
- max.intervals() gives the maximum fire interval.
- print.intervals() prints common fire-interval summary statistics.

median.intervals *Fire* intervals *median*

Description

Fire intervals median

Usage

```
## S3 method for class 'intervals'
median(x, ...)
```

Arguments

x	An intervals object.
...	Additional arguments passed to stats::median().

Value

Numeric or NA.

See Also

- intervals() to create a fire intervals object.
- mean.intervals() gets mean fire interval.
- quantile.intervals() get fit distribution quantiles.
- min.intervals() gives the minimum fire interval.
- max.intervals() gives the maximum fire interval.
- print.intervals() prints common fire-interval summary statistics.

---

min.intervals *Minimum interval in fire* intervals

Description

Minimum interval in fire intervals

Usage

```
## S3 method for class 'intervals'
min(x, ...)
```

Arguments

x	An intervals object.
...	Additional arguments passed to min().

Value

Numeric or NA.

See Also

- intervals() to create a fire intervals object.
- mean.intervals() gets median fire interval.
- median.intervals() gets median fire interval.
- quantile.intervals() get fit distribution quantiles.
- max.intervals() gives the maximum fire interval.
- print.intervals() prints common fire-interval summary statistics.

outer_type *Type of observation in the last (most recent) year of an* fhx *object*

Description

Type of observation in the last (most recent) year of an fhx object

Usage

```
outer_type(x)
```

Arguments

x	An fhx object.

Value

The a factor giving the type of observation in the last observation of x.

See Also

- inner_type() get observation type in inner-most year of fhx object.
- series_stats() basic statistics for series in an fhx object.

percent_scarred *Percent scarred time series for* fhx *object*

Description

Percent scarred time series for fhx object

Usage

```
percent_scarred(x, injury_event = FALSE)
```

Arguments

x	An fhx object.
injury_event	Boolean indicating whether years with injury events should be considered as scars. Default is FALSE.

Value

`data.frame` with four columns:

- "Year": The year.
- "NumRec": The number of recording trees.
- "NumScars": Number of fire scars and possibly fire injuries.
- "PercScarred": The proportion of scars (and possibly injuries) to non-scar/injury series in the year.

See Also

`series_stats()` basic statistics for series in an `fhx` object.

Examples

```
data("pgm")
percent_scarred(pgm)
```

---

pgm *Peggy Mesa fire-history data*

Description

An `fhx` object with fire-history data from Peggy Mesa.

Usage

```
pgm
```

Format

An `fhx` object with 41 series from 1555 to 2013 CE.

Source

Guiterman, Christopher H., Ellis Q. Margolis, and Thomas W. Swetnam. 2015. "Dendroecological Methods For Reconstructing High-Severity Fire In Pine-Oak Forests." Tree-Ring Research 71 (2): 67-77. doi:10.3959/1536-1098-71.2.67.

See Also

- pgm_meta Peggy Mesa metadata.
- pgm_pdsi PDSI time-series for Peggy Mesa site.

pgm_meta *Metadata for the Peggy Mesa fire-history dataset*

Description

A data frame with species and location information for the Peggy Mesa fire-history dataset (pgm).

Usage

```
pgm_meta
```

Format

A `data.frame` with 41 rows and 5 variables:

- "TreeID": Name of tree series.
- "SpeciesID": Abbreviated tree species.
- "Latitude": latitude of tree in decimal degrees.
- "Longitude": longitude of tree in decimal degrees.
- "Elevation": tree elevation in meters.

Source

Guiterman, Christopher H., Ellis Q. Margolis, and Thomas W. Swetnam. 2015. "Dendroecological Methods For Reconstructing High-Severity Fire In Pine-Oak Forests." Tree-Ring Research 71 (2): 67-77. doi:10.3959/1536-1098-71.2.67.

See Also

- pgm Peggy Mesa fire-history data.
- pgm_pdsi PDSI time-series for Peggy Mesa site.

pgm_pdsi *Reconstructed PDSI time series for the Peggy Mesa fire-history dataset*

Description

A tree-ring reconstructed Palmer Drought-Severity Index time series corresponding to the Peggy Mesa fire-history dataset (pgm) – specifically, the Jemez Mountains area (gridpoint 133). The reconstruction is from The North American Drought Atlas (Cook and Krusic 2004).

Usage

```
pgm_pdsi
```

Format

A data.frame with 2004 rows and 1 variables. Row names give the year for the reconstructed value:

- "RECON": The reconstructed PDSI series.

Source

Cook, E. R., and Krusic, P. J. (2004). The North American Drought Atlas. Retrieved September 13, 2017, from http://iridl.ldeo.columbia.edu/SOURCES/.LDEO/.TRL/.NADA2004/.pdsi-atlas.html

See Also

- pgm Peggy Mesa fire-history data.
- pgm_meta Peggy Mesa metadata.

---

plot.fhx *Plot an* fhx *object*

Description

Plot an fhx object

Usage

```
## S3 method for class 'fhx'
plot(...)
```

Arguments

...	Arguments passed on to plot_demograph().

See Also

plot_demograph() is what does the actual plotting.

Examples

```
data(lgr2)
plot(lgr2)

plot(lgr2, ylabels = FALSE, plot_legend = TRUE)

data(lgr2_meta)
# With color showing species.
plot(lgr2,
  color_group = lgr2_meta$SpeciesID,
  color_id = lgr2_meta$TreeID,
```

```
  plot_legend = TRUE
)
# With facets for each species.
plot(lgr2,
  facet_group = lgr2_meta$SpeciesID,
  facet_id = lgr2_meta$TreeID,
  plot_legend = TRUE
)

# Append annotation onto a ggplot object.
require(ggplot2)
p <- plot_demograph(lgr2,
  color_group = lgr2_meta$SpeciesID,
  color_id = lgr2_meta$TreeID
)
# Add transparent box as annotation to plot.
p + annotate("rect",
  xmin = 1750, xmax = 1805,
  ymin = 3.5, ymax = 13.5, alpha = 0.2
)
```

`plot.intervals` *Plot a fire* `intervals` *object*

Description

Plot a fire `intervals` object

Usage

```
## S3 method for class 'intervals'
plot(...)
```

Arguments

`...`	Arguments passed to `plot_intervals_dist()`.

See Also

`plot_intervals_dist()` plot `intervals` distributions.

Examples

```
data(pgm)
interv <- intervals(composite(pgm))

plot(interv, binwidth = 5)
```

plot.sea *Plot a* sea *object*

Description

Plot a sea object

Usage

```
## S3 method for class 'sea'
plot(...)
```

Arguments

...	Arguments passed on to plot_sealags().

See Also

plot_sealags() handles the plotting for this function.

Examples

```
## Not run:
# Read in the Cook and Krusic (2004; The North American Drought Atlas)
# reconstruction of Palmer Drought Severity Index (PDSI) for the Jemez
# Mountains area (gridpoint 133).
data(pgm_pdsi)

# Run SEA on Peggy Mesa (pgm) data
data(pgm)
pgm_comp <- composite(pgm)

pgm_sea <- sea(pgm_pdsi, pgm_comp)

plot(pgm_sea)

## End(Not run)
```

plot_demograph *Create an ggplot2 object for plotting fhx demographics*

Description

Create an ggplot2 object for plotting fhx demographics

Usage

```
plot_demograph(x, color_group, color_id, facet_group, facet_id,
  facet_type = "grid", ylabels = TRUE, yearlims = FALSE,
  composite_rug = FALSE, filter_prop = 0.25, filter_min_rec = 2,
  filter_min_events = 1, injury_event = FALSE, plot_legend = FALSE,
  event_size = c(Scar = 4, Injury = 2, `Pith/Bark` = 1.5),
  rugbuffer_size = 2, rugdivide_pos = 2)
```

Arguments

x	An fhx object, as from fhx()
color_group	Option to plot series with colors. This is a character vector or factor which corresponds to the series names given in color_id. Both color_group and color_id need to be specified. Default plot gives no color.
color_id	Option to plot series with colors. A character vector of series names corresponding to groups given in color_group. Every unique value in x series.names needs to have a corresponding color_group value. Both color_group and color_id need to be specified. Default plot gives no species colors.
facet_group	Option to plot series with faceted by a factor. A vector of factors or character vector which corresponds to the series names given in facet_id. Both facet_group and facet_id need to be specified. Default plot is not faceted.
facet_id	Option to plot series with faceted by a factor. A vector of series names corresponding to species names given in facet_group. Every unique values in x series.names needs to have a corresponding facet_group value. Both facet_group and facet_id need to be specified. Default plot is not faceted. Note that composite_rug, facet_group, and facet_id cannot be used in the same plot. You must choose facets or a composite rug.
facet_type	Type of ggplot2 facet to use, if faceting. Must be either "grid" or "wrap". Default is "grid". Note that composite_rug, facet_group, and facet_id cannot be used in the same plot. You must choose facets or a composite rug.
ylabels	Optional boolean to remove y-axis (series name) labels and tick marks. Default is TRUE.
yearlims	Option to limit the plot to a range of years. This is a vector with two integers. The first integer gives the lower year for the range while the second integer gives the upper year. The default is to plot the full range of data given by x.
composite_rug	A boolean option to plot a rug on the bottom of the plot. Default is FALSE. Note that composite_rug and facet_group, facet_id cannot be used in the same plot. You must choose facets or a composite rug.
filter_prop	The minimum proportion of fire events in recording series needed for fire event to be considered for composite. Default is 0.25.
filter_min_rec	The minimum number of recording series needed for a fire event to be considered for the composite. Default is 2 recording series.
filter_min_events	The minimum number of fire scars needed for a fire event to be considered for the composite. Default is 1. Fire injuries are included in this count if injury_event is TRUE.

injury_event	Boolean indicating whether injuries should be considered events. Default is FALSE.
plot_legend	A boolean option allowing the user to choose whether a legend is included in the plot or not. Default is FALSE.
event_size	An optional numeric vector that adjusts the size of fire event symbols on the plot. Default is c("Scar" = 4,"Injury" = 2,"Pith/Bark" = 1.5).
rugbuffer_size	An optional integer. If the user plots a rug, this controls the amount of buffer whitespace along the y-axis between the rug and the main plot. Must be >= 2.
rugdivide_pos	Optional integer if plotting a rug. Adjust the placement of the rug divider along the y-axis. Default is 2.

Value

A ggplot object for plotting or manipulation.

Examples

```
data(lgr2)
plot(lgr2)

plot(lgr2, ylabels = FALSE, plot_legend = TRUE)

data(lgr2_meta)
# With color showing species.
plot(lgr2,
  color_group = lgr2_meta$SpeciesID,
  color_id = lgr2_meta$TreeID,
  plot_legend = TRUE
)
# With facets for each species.
plot(lgr2,
  facet_group = lgr2_meta$SpeciesID,
  facet_id = lgr2_meta$TreeID,
  plot_legend = TRUE
)

# Append annotation onto a ggplot object.
require(ggplot2)
p <- plot_demograph(lgr2,
  color_group = lgr2_meta$SpeciesID,
  color_id = lgr2_meta$TreeID
)
# Add transparent box as annotation to plot.
p + annotate("rect",
  xmin = 1750, xmax = 1805,
  ymin = 3.5, ymax = 13.5, alpha = 0.2
)
```

plot_intervals_dist *Basic fire* intervals *distribution plot*

Description

Basic fire intervals distribution plot

Usage

```
plot_intervals_dist(x, binwidth = NULL)
```

Arguments

x	An intervals object, from intervals().
binwidth	The width of the bins. Can be specified as a numeric value or as a function that calculates width from unscaled x. Here, "unscaled x" refers to the original x values in the data, before application of any scale transformation. When specifying a function along with a grouping structure, the function will be called once per group. The default is to use bins bins that cover the range of the data. You should always override this value, exploring multiple widths to find the best to illustrate the stories in your data. The bin width of a date variable is the number of days in each time; the bin width of a time variable is the number of seconds.

Value

A ggplot object.

See Also

- intervals() to create a fire intervals object.
- mean.intervals() gets mean fire interval.
- median.intervals() gets median fire interval.
- quantile.intervals() get fit distribution quantiles.
- min.intervals() gives the minimum fire interval.
- max.intervals() gives the maximum fire interval.
- print.intervals() prints common fire-interval summary statistics.

plot_sealags *Basic SEA lag plot of* sea *object*

Description

Basic SEA lag plot of sea object

Usage

```
plot_sealags(x)
```

Arguments

x	A sea object.

Value

A ggplot object.

See Also

- sea() creates a sea object.
- print.sea() prints a pretty summary of a sea object.

Examples

```
## Not run:
# Read in the Cook and Krusic (2004; The North American Drought Atlas)
# reconstruction of Palmer Drought Severity Index (PDSI) for the Jemez
# Mountains area (gridpoint 133).
data(pgm_pdsi)

# Run SEA on Peggy Mesa (pgm) data
data(pgm)
pgm_comp <- composite(pgm)

pgm_sea <- sea(pgm_pdsi, pgm_comp)

plot(pgm_sea)

## End(Not run)
```

print.intervals *Print a fire* intervals *object*

Description

Print a fire intervals object

Usage

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

Arguments

x	An intervals object.
...	Additional arguments that are tossed.

See Also

intervals() to create a fire intervals object.

Examples

```
data(pgm)
interv <- intervals(composite(pgm))
print(interv)

# Note, you can also catch the printed table:
summary_stats <- print(interv)
```

print.sea *Print a* sea *object.*

Description

Print a sea object.

Usage

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

Arguments

x	A sea object.
...	Additional arguments that are tossed.

See Also

- sea() creates a sea object.
- plot_sealags() basic plot of sea object lags.

Examples

```
## Not run:
# Read in the Cook and Krusic (2004; The North American Drought Atlas)
# reconstruction of Palmer Drought Severity Index (PDSI) for the Jemez
# Mountains area (gridpoint 133).
target_url <- paste0(
  "http://iridl.ldeo.columbia.edu",
  "/SOURCES/.LDEO/.TRL/.NADA2004",
  "/pdsiatlashtml/pdsiwebdata/1050w_350n_133.txt"
)
pdsi <- read.table(target_url, header = TRUE, row.names = 1)
pdsi <- subset(pdsi, select = "RECON")

# Run SEA on Peggy Mesa (pgm) data
data(pgm)
pgm_comp <- composite(pgm)

pgm_sea <- sea(pdsi, pgm_comp)

# See basic results:
print(pgm_sea)

# Basic plot:
plot(pgm_sea)

## End(Not run)
```

---

quantile.intervals *Fit distribution quantiles to fire* intervals

---

Description

Fit distribution quantiles to fire intervals

Usage

```
## S3 method for class 'intervals'
quantile(x, q = c(0.125, 0.5, 0.875), ...)
```

Arguments

x	An intervals object.
q	Vector giving the desired quantiles.
...	Additional arguments passed to the quantile() method for the fit distribution.

See Also

- intervals() to create a fire intervals object.
- mean.intervals() gets median fire interval.
- median.intervals() gets median fire interval.
- quantile.intervals() get fit distribution quantiles.
- min.intervals() gives the minimum fire interval.
- max.intervals() gives the maximum fire interval.
- print.intervals() prints common fire-interval summary statistics.

Examples

```
data(pgm)
intervs <- intervals(composite(pgm))
quantile(intervs)

# Or you can pass in your own quantiles:
quantile(intervs, q = c(0.25, 0.5, 0.75))
```

---

read_fhx *Read FHX2 file and return an 'fhx" object*

Description

Read FHX2 file and return an 'fhx" object

Usage

```
read_fhx(fname, encoding, text)
```

Arguments

fname	Name of target FHX file. Needs to be in format version 2.
encoding	Encoding to use when reading the FHX file. The default is to use the system default in R.
text	Character string. If fname is not provided and text is, then data is read from text using a text connection.

Value

An fhx object, as returned by fhx().

See Also

- write_fhx() write an fhx object to a file.
- fhx() create an fhx object.
- as_fhx() cast data frame or similar object to an fhx object.

Examples

```
## Not run:
d <- read_fhx("afile.fhx")

## End(Not run)
```

`sample_depth` *Calculate the sample depth of an* `fhx` *object*

Description

Calculate the sample depth of an `fhx` object

Usage

```
sample_depth(x)
```

Arguments

`x`	An `fhx` object.

Value

A data frame containing the years and number of observations.

See Also

`series_stats()` basic statistics for series in an `fhx` object.

`sea` *Perform superposed epoch analysis*

Description

Perform superposed epoch analysis

Usage

```
sea(x, event, nbefore = 6, nafter = 4, event_range = TRUE,
  n_iter = 1000)
```

Arguments

x	A data frame climate reconstruction or tree-ring series with row names as years, and one numeric variable.
event	An numeric vector of event years for superposed epoch, such as fire years, or an fhx object with a single series as produced by composite().
nbefore	The number of lag years prior to the event year.
nafter	The number of lag years following the event year.
event_range	Logical. Constrain the time series to the time period of key events within the range of the x series. FALSE uses the entire series, ignoring the period of key events.
n_iter	The number of iterations for bootstrap resampling.

Details

Superposed epoch analysis (SEA) helps to evaluate fire-climate relationships in studies of tree-ring fire history. It works by compositing the values of an annual time series or climate reconstruction for the fire years provided (event) and both positive and negative lag years. Bootstrap resampling of the time series is performed to evaluate the statistical significance of each year's mean value. Users interpret the departure of the actual event year means from the simulated event year means. Note that there is no rescaling of the climate time series x.

The significance of lag-year departures from the average climate condition was first noted by Baisan and Swetnam (1990) and used in an organized SEA by Swetnam (1993). Since then, the procedure has been commonly applied in fire history studies. The FORTRAN program EVENT.exe was written by Richard Holmes and Thomas Swetnam (Holmes and Swetnam 1994) to perform SEA for fire history specifically. EVENT was incorporated in the FHX2 software by Henri Grissino-Mayer. Further information about SEA can be found in the FHAES user's manual, http://help.fhaes.org/.

sea() was originally designed to replicate EVENT as closely as possible. We have tried to stay true to their implementation of SEA, although multiple versions of the analysis exist in the climate literature and for fire history. The outcome of EVENT and sea should only differ slightly in the values of the simulated events and the departures, because random draws are used. The event year and lag significance levels should match, at least in the general pattern.

Our SEA implementation borrowed from dplR::sea() function in how it performs the bootstrap procedure, but differs in the kind of output provided for the user.

Value

A sea object containing. This contains:

- "event_years": a numeric vector of event years.
- "actual": a data.frame summary of the actual events.
- "random": a data.frame summary of the bootstrapped events.
- "departure": a data.frame summary of the departures of actual from bootstrapped events.
- "simulated": a full 2D matrix of the bootstrapped-values across lags.
- "observed": a ful 2D matrix of "actual" events across lags.

References

Baisan and Swetnam 1990, Fire history on desert mountain range: Rincon Mountain Wilderness, Arizona, U.S.A. Canadian Journal of Forest Research 20:1559-1569.

Bunn 2008, A dendrochronology program library in R (dplR), Dendrochronologia 26:115-124

Holmes and Swetnam 1994, EVENT program description

Swetnam 1993, Fire history and climate change in giant sequoia groves, Science 262:885-889.

See Also

- `plot_sealags()` plots sea lags and their statistical significance.
- `print.sea()` prints a pretty summary of sea objects.
- `composite()` creates fire composites, a common input to `sea()`.

Examples

```
## Not run:
# Read in the Cook and Krusic (2004; The North American Drought Atlas)
# reconstruction of Palmer Drought Severity Index (PDSI) for the Jemez
# Mountains area (gridpoint 133).
target_url <- paste0(
  "http://iridl.ldeo.columbia.edu",
  "/SOURCES/.LDEO/.TRL/.NADA2004",
  "/pdsiatlashtml/pdsiwebdata/1050w_350n_133.txt"
)
pdsi <- read.table(target_url, header = TRUE, row.names = 1)
pdsi <- subset(pdsi, select = "RECON")

# Run SEA on Peggy Mesa (pgm) data
data(pgm)
pgm_comp <- composite(pgm)

pgm_sea <- sea(pdsi, pgm_comp)

# See basic results:
print(pgm_sea)

# Basic plot:
plot(pgm_sea)

## End(Not run)
```

---

series_mean_interval *Calculate quick mean fire interval of an* fhx *object with single series*

Description

You really should be using `intervals()`.

Usage

```
series_mean_interval(x, injury_event = FALSE)
```

Arguments

x	An fhx object with a single series.
injury_event	Boolean indicating whether injuries should be considered event.

Value

The mean fire interval observed x.

See Also

- intervals() Proper way to do fire-interval analysis of fhx object.
- series_stats() basic statistics for series in an fhx object.

---

series_names *Get* fhx *series names*

Description

Get fhx series names

Usage

```
series_names(x)
```

Arguments

x	An fhx object.

Value

A character vector or NULL.

See Also

- series_names() get all the series in an fhx object.
- get_year() subset an fhx object to select years.
- year_range() get earliest and latest year in an fhx object.
- get_series() subset an fhx object to select series.
- get_event_years() gets years for various events in an fhx object.
- count_event_position() count the number of different events in an fhx object.
- yearly_recording() count the number of "recording" events in each year of an fhx object.
- series_stats() basic summary stats for an fhx object.

Examples

```
data(lgr2)
series_names(lgr2)
```

`series_stats` *Generate series-level descriptive statistics for* `fhx` *object*

Description

Generate series-level descriptive statistics for `fhx` object

Usage

```
series_stats(x, func_list = list(first = first_year, last = last_year,
  years = count_year_span, inner_type = inner_type, outer_type =
  outer_type, number_scars = count_scar, number_injuries = count_injury,
  recording_years = count_recording, mean_interval = series_mean_interval))
```

Arguments

`x`	An `fhx` object.
`func_list`	A list of named functions that will be run on each series in the `fhx` object. The list name for each function is the corresponding column name in the output data frame.

Value

A `data.frame` containing series-level statistics.

See Also

- `fhx()` creates an `fhx` object.
- `as_fhx()` casts data frame into an `fhx` object.
- `first_year()` gets earliest year in an `fhx` object.
- `last_year()` gets latest year in an `fhx` object.
- `count_year_span()` counts the year span of an `fhx` object.
- `inner_type()` gets "rec_type" for inner event of an `fhx` object.
- `outer_type()` get "rec_type" for outside event of an `fhx` object.
- `count_scar()` counts scars in an `fhx` object.
- `count_injury()` counts injuries in an `fhx` object.
- `count_recording()` counts recording years in `fhx` object.
- `series_mean_interval()` quickly estimates mean fire-interval of `fhx` object.
- `sample_depth()` gets sample depth of an `fhx` object.

- summary.fhx() brief summary of an fhx object.
- composite() create a fire composite from an fhx object.
- intervals() get fire intervals analysis from composite.
- sea() superposed epoch analysis.

Examples

```
data(lgr2)
series_stats(lgr2)

# You can create your own list of statistics to output. You can also create
# your own functions:
flist <- list(
  n = count_year_span,
  xbar_interval = function(x) mean_interval(x, injury_event = TRUE)
)
sstats <- series_stats(lgr2)
head(sstats)
```

---

sort.fhx *Sort the series names of* fhx *object by the earliest or latest year*

Description

Sort the series names of fhx object by the earliest or latest year

Usage

```
## S3 method for class 'fhx'
sort(x, decreasing = FALSE, sort_by = "first_year", ...)
```

Arguments

x	An fhx object to sort.
decreasing	Logical. Decreasing sorting? Defaults to FALSE.
sort_by	Either "first_year" or "last_year". Designates the inner or outer year for sorting. Defaults to "first_year"
...	Additional arguments that fall off the face of the universe.

Value

A copy of x with reordered series.

See Also

- fhx() constructs an fhx object.
- as_fhx() casts data frame-like object into an fhx object.
- series_names() get all the series in an fhx object.
- delete() remove observations from an fhx object.
- +.fhx() concatenate multiple fhx objects together.

Examples

```
data(lgr2)
plot(sort(lgr2, decreasing = TRUE))
plot(sort(lgr2, sort_by = "last_year"))
```

---

summary.fhx *Summary of* fhx *object*

Description

Summary of fhx object

Usage

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

Arguments

object	An fhx object.
...	Additional arguments that are tossed out.

Value

A summary.fhx object.

See Also

series_stats() basic statistics for series in an fhx object.

write_fhx *Write an* fhx *object to a new FHX2 file*

Description

Write an fhx object to a new FHX2 file

Usage

```
write_fhx(x, fname = "")
```

Arguments

x	An fhx object.
fname	Output filename.

See Also

- write.csv() to write a CSV file. Also works on fhx objects.
- read_fhx() to read an FHX2 file.

Examples

```
## Not run:
data(lgr2)
write_fhx(lgr2, "afile.fhx")

## End(Not run)
```

yearly_recording *Count the number of recording series for each year in an* fhx *object*

Description

Count the number of recording series for each year in an fhx object

Usage

```
yearly_recording(x, injury_event = FALSE)
```

Arguments

x	An fhx object.
injury_event	Boolean indicating whether injuries should be considered events. Default is FALSE.

Value

A data frame with columns giving the year and recording events count.

Examples

```
data(lgr2)
yearly_recording(lgr2)
```

---

year_range *Range of years in an* fhx *object*

Description

Range of years in an fhx object

Usage

```
year_range(x)
```

Arguments

x	An fhx object.

Value

A numeric vector or NULL.

See Also

- series_names() get all the series in an fhx object.
- get_year() subset an fhx object to select years.
- get_series() subset an fhx object to select series.
- get_event_years() gets years for various events in an fhx object.
- count_event_position() count the number of different events in an fhx object.
- yearly_recording() count the number of "recording" events in each year of an fhx object.
- series_stats() basic summary stats for an fhx object.

Examples

```
data(lgr2)
year_range(lgr2)
```

Index