Package ‘burnr’

March 1, 2022

Title Forest Fire History Analysis
Version 0.6.1
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
Importsforcats, ggplot2, MASS, plyr, reshape2, rlang, stringr, tidyR
RoxygenNote 7.1.2
VignetteBuilder knitr
NeedsCompilation no
Author Steven Malevich [aut, cre] (<https://orcid.org/0000-0002-4752-8190>), Christopher Guiterman [aut, ctb] (<https://orcid.org/0000-0002-9706-9332>), Ellis Margolis [aut] (<https://orcid.org/0000-0002-0595-9005>)
Maintainer Steven Malevich <sbmalev@gmail.com>
Repository CRAN
Date/Publication 2022-03-01 10:10:06 UTC

R topics documented:

+.fhx .................................................... 3
as.fhx .................................................. 4
as_fhx .................................................. 4
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>composite</td>
</tr>
<tr>
<td>count_event_position</td>
</tr>
<tr>
<td>count_injury</td>
</tr>
<tr>
<td>count_recording</td>
</tr>
<tr>
<td>count_scar</td>
</tr>
<tr>
<td>count_year_span</td>
</tr>
<tr>
<td>delete</td>
</tr>
<tr>
<td>fhx</td>
</tr>
<tr>
<td>first_year</td>
</tr>
<tr>
<td>get_event_years</td>
</tr>
<tr>
<td>get_series</td>
</tr>
<tr>
<td>get_year</td>
</tr>
<tr>
<td>inner_type</td>
</tr>
<tr>
<td>intervals</td>
</tr>
<tr>
<td>is.fhx</td>
</tr>
<tr>
<td>is.intervals</td>
</tr>
<tr>
<td>is.sea</td>
</tr>
<tr>
<td>is_fhx</td>
</tr>
<tr>
<td>is_intervals</td>
</tr>
<tr>
<td>is_sea</td>
</tr>
<tr>
<td>last_year</td>
</tr>
<tr>
<td>lgr2</td>
</tr>
<tr>
<td>lgr2_meta</td>
</tr>
<tr>
<td>make_rec_type</td>
</tr>
<tr>
<td>max.intervals</td>
</tr>
<tr>
<td>mean.intervals</td>
</tr>
<tr>
<td>median.intervals</td>
</tr>
<tr>
<td>min.intervals</td>
</tr>
<tr>
<td>outer_type</td>
</tr>
<tr>
<td>percent_scarred</td>
</tr>
<tr>
<td>pgm</td>
</tr>
<tr>
<td>pgm_meta</td>
</tr>
<tr>
<td>pgm_pdsi</td>
</tr>
<tr>
<td>plot.fhx</td>
</tr>
<tr>
<td>plot.intervals</td>
</tr>
<tr>
<td>plot.sea</td>
</tr>
<tr>
<td>plot_demograph</td>
</tr>
<tr>
<td>plot_intervals_dist</td>
</tr>
<tr>
<td>plot_sealags</td>
</tr>
<tr>
<td>pme</td>
</tr>
<tr>
<td>pmr</td>
</tr>
<tr>
<td>pmw</td>
</tr>
<tr>
<td>print.intervals</td>
</tr>
<tr>
<td>print_sea</td>
</tr>
<tr>
<td>quantile.intervals</td>
</tr>
<tr>
<td>read_fhx</td>
</tr>
<tr>
<td>sample_depth</td>
</tr>
<tr>
<td>sea</td>
</tr>
</tbody>
</table>
Description

Concatenate or combine two fhx objects

Usage

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

Arguments

- `a`: An fhx object.
- `b`: The fhx object to 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

```r
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)
composite

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)

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)
   plot(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)
**Usage**

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

**Arguments**

- **x**
  - An fhx object.

- **injury_event**
  - Optional boolean indicating whether injuries should be considered an "event". Default is FALSE.

- **position**
  - Deprecated. This allowed users to specify which intra-ring positions to include in the summary output table. The default counts all types of event positions.

- **drop_unknown**
  - Boolean. Defaults to FALSE. If TRUE will remove the "unknown_fs" and/or "unknown_fi" from rec_type.

- **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 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**

```r
data(pgm)
count_event_position(pgm)
# As above, but considering injuries to be a type of event.
count_event_position(pgm, injury_event = TRUE)
# Often we only quantify known intra-ring positions.
# Remove the "unknown_fs" and/or "unknown_fi" with
count_event_position(pgm, drop_unknown = TRUE)
# Using custom "groupby" args in a named list, as
grpplist <- list(
  foo = c("dormant_fs", "early_fs"),
  bar = c("middle_fs", "late_fs")
)
count_recording

)  
count_event_position(pgm, groupby = grplist)
# Note that if a position in the groupby list is
# not included in rec_type, forcats::fct_count()
# will throw a flag for an "Unknown levels in 'f':"
**count_scar**

**Value**

The number of recording events in \( x \).

**See Also**

series_stats() basic statistics for series in an \( \text{fhx} \) object.

---

| count_scar       | Number of scar events in an \( \text{fhx} \) object |

**Description**

Number of scar events in an \( \text{fhx} \) object

**Usage**

count_scar(x)

**Arguments**

- \( x \) An \( \text{fhx} \) object.

**Value**

The number of fire scar events in \( x \)

**See Also**

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

---

| count_year_span   | Number of years of an \( \text{fhx} \) object |

**Description**

Number of years of an \( \text{fhx} \) object

**Usage**

count_year_span(x)

**Arguments**

- \( x \) An \( \text{fhx} \) 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.

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

```r
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

```r
first_year(x)
```

Arguments

- `x` An fhx object.

Value

The minimum or first year of series in `x`.  

get_event_years

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)

## Not run:
# Subsetting before/after a specific year requires a
call to year_range(). For example, to extract all observations
# prior to 1900, use
get_year(lgr2, year_range(lgr2)[1]:1900)

## End(Not run)
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**

- **Description**
  
  Alias to `is_fhx()`
  
- **Usage**
  
  ```r
  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**
  
  ```r
  data(lgr2)
  is_fhx(lgr2)
  ```

---

**is.intervals**

- **Description**
  
  Alias to `is_intervals()`

- **Usage**
  
  ```r
  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.
is_intervals

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_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**

```r
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**

```r
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**

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

x An intervals object.
...

Additional arguments passed to \texttt{max()}.

Value

Numeric or NA.

See Also

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

---

**mean.intervals**

*Fire intervals arithmetic mean*

Description

Fire intervals arithmetic mean

Usage

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

Arguments

x An intervals object.
...

Additional arguments passed to \texttt{mean()}.

Value

Numeric or NA.

See Also

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

Description
Minimum interval in fire intervals

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

Arguments

x  An intervals object.

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.
**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**

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

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

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


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**


**See Also**

- `pgm` Peggy Mesa fire-history data.
- `pgm_meta` Peggy Mesa metadata.
## plot.fhx

### Description

Plot an fhx object

### Usage

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

### Arguments

...  
Arguments passed on to `plot_demograph()`.

### See Also

`plot_demograph()` is what does the actual plotting.

### Examples

```r
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,
   ...)  
```
plot.intervals

Description
Plot a fire intervals object

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

Arguments
...

See Also
plot_intervals_dist() plot intervals distributions.

Examples
data(pgm)
interv <- intervals(composite(pgm))
plot(interv, binwidth = 5)

plot.sea

Description
Plot a sea object

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

Arguments
...

Arguments passed on to plot_sealags().
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
)

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)
**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.

- **ylables**
  - 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`, `facet_group`, and `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 $\geq 2$.

**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**

```r
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
**plot_intervals_dist**

**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 the number of bins in `bins`, covering 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

```r
## 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)

data(pgm)
pgm_comp <- composite(pgm)

pgm_sea <- sea(pgm_pdsi, pgm_comp)

plot(pgm_sea)
## End(Not run)
```

---

pme  

*Pajarito Mountain East fire-history data*

Description

An fhx object with fire-history data.

Usage

pme

Format

An fhx object with 17 series from 1702 to 1993 CE.

Source

https://www1.ncdc.noaa.gov/pub/data/paleo/firehistory/firescar/northamerica/uspme001.fhx
pmr

**Pajarito Mountain Ridge fire-history data**

**Description**

An fhx object with fire-history data.

**Usage**

`pmr`

**Format**

An fhx object with 23 series from 1626 to 1993 CE.

**Source**

[https://www1.ncdc.noaa.gov/pub/data/paleo/firehistory/firescar/northamerica/uspmr001.fhx](https://www1.ncdc.noaa.gov/pub/data/paleo/firehistory/firescar/northamerica/uspmr001.fhx)

---

pmw

**Pajarito Mountain West fire-history data**

**Description**

An fhx object with fire-history data.

**Usage**

`pmw`

**Format**

An fhx object with 11 series from 1617 to 1993 CE.

**Source**

[https://www1.ncdc.noaa.gov/pub/data/paleo/firehistory/firescar/northamerica/uspmw001.fhx](https://www1.ncdc.noaa.gov/pub/data/paleo/firehistory/firescar/northamerica/uspmw001.fhx)
print.intervals  

*Print a fire intervals object*

---

**Description**

Print a fire intervals object

**Usage**

```r
## 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**

```r
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**

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

**Arguments**

- `x` A sea object.
- `...` Additional arguments that are tossed.
quantile.intervals

See Also

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

Examples

```r
## 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",
  "/pdsi atlas html/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

```r
## 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 \texttt{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 \texttt{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 \texttt{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/.

\texttt{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 \texttt{dplR::sea()} function in how it performs the bootstrap procedure, but differs in the kind of output provided for the user.

Value

A \texttt{sea} object containing. This contains:

- "event\_years": a numeric vector of event years.
- "actual": a \texttt{data.frame} summary of the actual events.
- "random": a \texttt{data.frame} summary of the bootstrapped events.
- "departure": a \texttt{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.
series_mean_interval

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

Description

You really should be using intervals().

References


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

Holmes and Swetnam 1994, EVENT program description


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

```r
## 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)
```
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

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

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

**Description**

Generate series-level descriptive statistics for fhx object

**Usage**

```r
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.
sort.fhx

- `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

```r
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**

```r
# 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

- `f hx()` constructs an fhx object.
- `as_fh x()` 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.
- `+.fh x()` concatenate multiple fhx objects together.

Examples

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

Summary of fhx object

### Description

Summary of fhx object

### Usage

```r
## S3 method for class 'fh x'
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 FH2 file**

**Description**

Write an fhx object to a new FH2 file

**Usage**

```r
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 FH2 file.

**Examples**

```r
# 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**

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

**Arguments**

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

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

* datasets
  lgr2, 22
  lgr2_meta, 22
  pgm, 28
  pgm_meta, 28
  pgm_pdsi, 29
  pme, 36
  pmr, 37
  pmw, 37
  .fhx, 3
  .fhx(), 12, 18, 20, 47
  as.factor(), 11
  as.fhx, 4
  as.numeric(), 11
  as_fhx(), 4, 6, 10, 12, 18, 20, 23, 40, 45, 47
  composite, 5
  composite(), 12, 16, 17, 42, 43, 46
  count_event_position, 6
  count_event_position(), 6, 13, 44, 49
  count_injury, 8
  count_injury(), 9, 45
  count_recording, 8
  count_recording(), 46
  count_scar, 9
  count_scar(), 8, 45
  count_year_span, 9
  count_year_span(), 45
  delete, 10
  delete(), 3, 14, 15, 47
  fhx, 11
  fhx(), 4–6, 10, 18, 20, 23, 33, 40, 45, 47
  first_year, 12
  first_year(), 10, 21, 45
  get_event_years, 13
  get_event_years(), 6, 7, 10, 13, 15, 44, 49
  get_series, 14
  get_series(), 3, 10, 13, 15, 44, 49
  get_year, 15
  get_year(), 10, 13, 14, 44, 49
  inner_type, 16
  inner_type(), 26, 45
  intervals, 16
  intervals(), 6, 12, 19, 20, 24–26, 35, 38, 40, 43, 44, 46
  is.fhx, 18
  is.intervals, 18
  is_sea, 19
  is_fhx, 19
  is_fhx(), 4, 5, 12, 18
  is_intervals, 20
  is_intervals(), 18
  is_sea, 21
  is_sea(), 19
  last_year, 21
  last_year(), 10, 13, 45
  lgr2, 22, 22
  lgr2_meta, 22, 22
  make_rec_type, 23
  make_rec_type(), 4, 5, 11, 12
  MASS::fitdistr(), 17
  max(), 24
  max.intervals, 23
  max.intervals(), 17, 24–26, 35, 40
  mean(), 24
  mean.intervals, 24
  mean.intervals(), 17, 24–26, 35, 40
  median.intervals, 25
  median.intervals(), 17, 24, 26, 35, 40
  min(), 26
  min.intervals, 25
  min.intervals(), 17, 24, 25, 35, 40
  outer_type, 26
INDEX

outer_type(), 16, 45
percent_scarred, 27
pgm, 28, 29
pgm_meta, 28, 29
pgm_pdsi, 28, 29
plot.fhx, 30
plot.intervals, 31
plot.sea, 31
plot_demograph, 32
plot_intervals_dist(), 17, 31
plot_sealags, 35
plot_sealags(), 31, 32, 39, 43
pme, 36
pmr, 37
pmw, 37
print.intervals, 38
print.intervals(), 17, 24–26, 35, 40
print.sea, 38
print.sea(), 36, 43
quantile(), 39
quantile.intervals, 39
quantile.intervals(), 17, 24–26, 35, 40
read_fhx, 40
read_fhx(), 12, 48
sample_depth, 41
sample_depth(), 46
sea, 41
sea(), 6, 12, 19, 21, 36, 39, 42, 43, 46
series_mean_interval, 43
series_mean_interval(), 46
series_names, 44
series_names(), 3, 10, 13, 14, 44, 47, 49
series_stats, 45
series_stats(), 6–10, 12, 13, 16, 21, 26, 27, 41, 44, 47, 49
sort.fhx, 46
sort.fhx(), 3, 12
stats::ks.test(), 17
stats::median(), 25
stats::shapiro.test(), 17
summary.fhx, 47
summary.fhx(), 46
write.csv(), 48
write_fhx, 48
write_fhx(), 12, 40
year_range, 49
year_range(), 10, 13, 15, 44
yearly_recording, 48
yearly_recording(), 6, 7, 13, 44, 49