Package ‘sapfluxnetr’

January 25, 2023

Title Working with ‘Sapfluxnet’ Project Data
Version 0.1.4
Description Access, modify, aggregate and plot data from the 'Sapfluxnet' project (<http://sapfluxnet.creaf.cat>), the first global database of sap flow measurements.
Depends R (>= 3.5.0)
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports assertthat, dplyr, furrr, ggplot2, glue, lubridate, magrittr, methods, purrr, rlang, stats, stringr, tibble, tidyr
RoxygenNote 7.2.3
Collate 'data.R' 'getters.R' 'helpers.R' 'imports.R' 'metrics.R' 'sfn_data_classes.R' 'sfn_data_generics.R' 'sfn_data_methods.R' 'sfn_dplyr.R' 'visualizations.R'
Suggests future, knitr, remotes, rmarkdown, testthat, xtable
Config/testthat/edition 3
Config/testthat/parallel true
VignetteBuilder knitr
URL https://github.com/sapfluxnet/sapfluxnetr
BugReports https://github.com/sapfluxnet/sapfluxnetr/issues
NeedsCompilation no
Author Victor Granda [aut, cre] (<https://orcid.org/0000-0002-0469-1991>), Rafael Poyatos [aut] (<https://orcid.org/0000-0003-0521-2523>), Victor Flo [aut] (<https://orcid.org/0000-0003-1908-4577>), Jacob Nelson [ctb] (<https://orcid.org/0000-0002-4663-2420>), Sapfluxnet Core Team [cph]
Maintainer Victor Granda <victorgrandagarcia@gmail.com>
Repository CRAN
Date/Publication 2023-01-25 15:30:02 UTC
\textbf{R topics documented:}

ARG_MAZ ......................................................... 3
ARG_TRE ......................................................... 3
AUS_CAN_ST2_MIX ............................................... 4
data_coverage ............................................... 4
describe_md_variable ....................................... 5
diurnal_centroid ............................................. 6
filter_sites_by_md ........................................... 7
get_timezone .................................................... 8
initialize.sfn_data-method ................................ 9
initialize.sfn_data_multi-method ......................... 10
metrics ........................................................ 11
metrics_tidyfier .............................................. 16
norm_diurnal_centroid ....................................... 18
read_sfn_data ................................................ 19
read_sfn_metadata ............................................ 20
sfn_data-class ................................................ 21
sfn_data_multi-class ........................................ 21
sfn_data_multi_validity ..................................... 22
sfn_data_validity ............................................. 22
sfn_filter ....................................................... 22
sfn_get.generics ............................................. 24
sfn_get.methods .............................................. 25
sfn_metadata_ex .............................................. 26
sfn_metrics ..................................................... 27
sfn_multi_get.methods ....................................... 30
sfn_mutate ..................................................... 31
sfn_mutate_at ................................................ 32
sfn_plot ......................................................... 34
sfn_replacement.generics .................................. 36
sfn_replacement.methods ................................... 37
sfn_sites_in_folder ......................................... 38
sfn_vars_to_filter ........................................... 39
show,sfn_data-method ....................................... 40
show,sfn_data_multi-method ................................. 40
summarise_by_period ....................................... 41
\%\% .......................................................... 42

Index 44
ARG_MAZ

ARG_MAZ sapfluxnet site

Description
Example site for package usage demonstration based on ARG_MAZ

Usage
ARG_MAZ

Format
An sfn_data class object with the data and metadata for ARG_MAZ site

Examples

data('ARG_MAZ', package = 'sapfluxnetr')
ARG_MAZ

ARG_TRE

ARG_TRE sapfluxnet site

Description
Example site for package usage demonstration based on ARG_TRE

Usage
ARG_TRE

Format
An sfn_data class object with the data and metadata for ARG_TRE site

Examples

data('ARG_TRE', package = 'sapfluxnetr')
ARG_TRE
**AUS_CAN_ST2_MIX**  
*AUS_CAN_ST2_MIX* sapfluxnet site

**Description**
Example site for package usage demonstration based on AUS_CAN_ST2_MIX

**Usage**

```
AUS_CAN_ST2_MIX
```

**Format**

An sfn_data class object with the data and metadata for AUS_CAN_ST2_MIX site

**Examples**

```
data('AUS_CAN_ST2_MIX', package = 'sapfluxnetr')
AUS_CAN_ST2_MIX
```

---

**data_coverage**

data coverage

**Description**

helper for sfn_metrics

**Usage**

```
data_coverage(x, timestep, period_minutes)
```

**Arguments**

- `x` a vector, usually a variable in the sapflow or environmental data.
- `timestep` numeric value with the timestep in minutes
- `period_minutes` numeric value with the period in minutes

**Details**

This helper function calculates the coverage percentage in a vector, and is designed to be used inside a dplyr summarise statement. It calculates the coverage as the percentage of no NAs in the expected length of the summarising period stated by the timestep.
describe_md_variable

**Value**

a single value (numeric) with the percentage of coverage for that variable

**Examples**

```r
# data for one day, 60 minutes timestep (24 values) with a 75% of coverage
x <- rep(c(1,2,3,NA), 6)
data_coverage(x, 60, 1440) # 75
```

describe_md_variable  Detailed description of metadata variables

**Description**

describe_md_variable prints in console a detailed description for the requested variable. Useful to know which values to filter or in which units the variables are.

**Usage**

```r
describe_md_variable(variable)
```

**Arguments**

- `variable`  A character with the name of the variable

**Value**

Nothing, prints information to console

**Examples**

```r
# info about the method used to measure sapflow (pl_sens_meth)
describe_md_variable('pl_sens_meth')
```
Diurnal centroid calculation

Description
Calculate the diurnal centroid for sapflow variables

Usage

diurnal_centroid(variable)

Arguments

variable A numeric vector containing the sapflow values for a day at a regular intervals. Missing values are allowed but not recommended

Details
The code for this function has been kindly provided by Jacob Nelson in python (see https://github.com/jnelson18/FluxnetTools/blob/master/FileS3.py) and has been translated to a tidy data philosophy in R to be used inside a summarise statement.

Value
A numeric value with the diurnal centroid value (0 to 24 h)

Diurnal centroid algorithm
Given a continuous subdaily sapflow values at regular intervals \( V = x_1, \ldots, x_n \) to obtain the diurnal centroid each value is multiplied by its interval index and summed up and divided by the sum of the values for the day and finally the value is normalized to 24h:

\[
\sum x_1 \cdot 1, x_2 \cdot 2, \ldots, x_n \cdot n / \sum x_1, x_2, \ldots, x_n \cdot (24/n)
\]

With even values for all the intervals (i.e. 100 for all), centroid converges to 12h at more than 1000 intervals per day. With only 48 (half hourly measurements) centroid converges to 12.25h and with 24 intervals (hourly measurements) centroid converges to 12.5h. So, using diurnal centroid value in half hourly datasets or above can have a considerable error associated.

Author(s)
Jacob Nelson & Víctor Granda
Examples

```r
# dplyr
library(dplyr)

# check convergence to 12h:
diurnal_centroid(rep(1, 1000)) # 12.012 h
diurnal_centroid(rep(10000, 1000)) # 12.012 h, variable scale not affects calculation

# sapflow diurnal centroid
data('ARG_TRE', package = 'sapfluxnetr')
sfn_metrics(
  ARG_TRE,
  period = '1 day',
  .funs = list(~ diurnal_centroid(.),
                ~ data_coverage(., timestep, period_minutes)),
  solar = FALSE,
  interval = 'general'
)
```

---

**filter_sites_by_md**  
Filter the sites by metadata variable values

**Description**

The `filter_sites_by_md` function takes logical expressions for the metadata variables (i.e. `pl_sens_meth == 'HR'`), and list the sites that met the criteria from those supplied.

**Usage**

```r
filter_sites_by_md(sites, metadata, ..., .join = c("and", "or"))
```

**Arguments**

- `sites` character vector with the sites codes to filter, generally the result of `sfn_sites_in_folder`
- `metadata` metadata tbl object, usually the result of `read_sfn_metadata`
- `...` Logical expressions for the metadata variables, as in `filter`
- `.` join Character indicating how to filter the sites, see details.

**Details**

The `.join` argument indicates how sites must be filtered between metadata classes. 'and' indicates only sites meeting all conditions for all metadata classes are returned. 'or' indicates all sites meeting any condition between classes are returned. For two or more filters of the same metadata class, they are combined as 'and'.
Value

A character vector with the sites fulfilling the premises

Examples

```r
# Let's access the data in "folder". This typically is the folder where the
# sapflow data at the desired unit level is (i.e. "RData/plant"), but in this
# example we will create a temporal folder with some data to test the function
directory <- tempdir()
save(ARG_TRE, file = file.path(directory, 'ARG_TRE.RData'))
save(ARG_MAZ, file = file.path(directory, 'ARG_MAZ.RData'))
save(AUS_CAN_ST2_MIX, file = file.path(directory, 'AUS_CAN_ST2_MIX.RData'))

# we need the metadata and the site names
metadata <- read_sfn_metadata(folders = directory, .write_cache = TRUE)
sites <- sfn_sites_in_folder(folders)

# Filter by Heat Ratio method
filter_sites_by_md(
  pl_sens_meth == 'HR', sites = sites, metadata = metadata)

# Both, Heat Ratio and Heat Dissipation
filter_sites_by_md(
  pl_sens_meth %in% c('HR', 'HD'),
  sites = sites, metadata = metadata)

# more complex, Heat Ratio method AND Woodland/Shrubland biome
filter_sites_by_md(
  pl_sens_meth == 'HR',
  si_biome == 'Woodland/Shrubland',
  sites = sites, metadata = metadata,
  .join = 'and' # default)

# join = 'or' returns sites that meet any condition
filter_sites_by_md(
  pl_sens_meth == 'HR',
  si_biome == 'Woodland/Shrubland',
  sites = sites, metadata = metadata,
  .join = 'or')
```

Description

Obtain the site timezone from a sfn_data/sfn_data_multi object
initialize, sfn_data-method

Usage

get_timezone(sfn_data)

Arguments

sfn_data An sfn_data or sfn_data_multi object

Value

a character with the site timezone

Examples

# timezone of ARG_TRE site
get_timezone(ARG_TRE)

initialize, sfn_data-method

Initialize method for sfn_data

Description

Initialize an sfn_data object

Usage

## S4 method for signature 'sfn_data'
initialize(
  .Object,
  sapf_data,
  env_data,
  sapf_flags,
  env_flags,
  si_code,
  timestamp,
  solar_timestamp,
  site_md,
  stand_md,
  species_md,
  plant_md,
  env_md
)
**Arguments**

- `.Object` sfn_data object to create
- `sapf_data` A tibble (or any object coercible to one) with the sapf_data (without the TIMES-TAMP variable)
- `env_data` A tibble (or any object coercible to one) with the env_data (without the TIMES-TAMP variable)
- `sapf_flags` A tibble (or any object coercible to one) with the same dimensions of sapf_data with the flag info for each tree/TIMESTAMP combination
- `env_flags` A tibble (or any object coercible to one) with the same dimensions of env_data with the flag info for each env_var/TIMESTAMP combination
- `si_code` A character vector of length one indicating the site code
- `timestamp` A POSIXct vector of length nrow(sapf_data) with the timestamp
- `solar_timestamp` A POSIXct vector of length nrow(sapf_data) with the solar timestamp
- `site_md` A tibble (or any object coercible to one) containing the site metadata
- `stand_md` A tibble (or any object coercible to one) containing the stand metadata
- `species_md` A tibble (or any object coercible to one) containing the species metadata
- `plant_md` A tibble (or any object coercible to one) containing the plant metadata
- `env_md` A tibble (or any object coercible to one) containing the env metadata

---

**Description**

Initialize an sfn_data_multi object

**Usage**

```r
## S4 method for signature 'sfn_data_multi'
initialize(.Object, ...)
```

**Arguments**

- `.Object` sfn_data_multi object to create
- `...` sfn_data elements
Description
This set of functions returns a complete set of statistics for a site (using \texttt{sfn_data}) or several sites (using \texttt{sfn_data_multi}).

Usage
\begin{verbatim}
daily_metrics(
  sfn_data,
  solar = TRUE,
  probs = c(0.95),
  tidy = FALSE,
  metadata = NULL,
  ...
)

monthly_metrics(
  sfn_data,
  solar = TRUE,
  probs = c(0.95),
  tidy = FALSE,
  metadata = NULL,
  ...
)

nightly_metrics(
  sfn_data,
  period = c("1 day", "1 month"),
  solar = TRUE,
  int_start = 20,
  int_end = 6,
  probs = c(0.95),
  tidy = FALSE,
  metadata = NULL,
  ...
)

daylight_metrics(
  sfn_data,
  period = c("1 day", "1 month"),
  solar = TRUE,
  int_start = 6,
  int_end = 20,
  probs = c(0.95),
  ...
tidy = FALSE,
metadata = NULL,
... 
)

predawn_metrics(
  sfn_data,
  period = c("1 day", "1 month"),
  solar = TRUE,
  int_start = 4,
  int_end = 6,
  probs = c(0.95),
  tidy = FALSE,
  metadata = NULL,
  ... 
)

midday_metrics(
  sfn_data,
  period = c("1 day", "1 month"),
  solar = TRUE,
  int_start = 11,
  int_end = 13,
  probs = c(0.95),
  tidy = FALSE,
  metadata = NULL,
  ... 
)

Arguments

- **sfn_data**: `sfn_data` or `sfn_data_multi` object to obtain the metrics from
- **solar**: Logical indicating if the solarTIMESTAMP must be used instead of the site local TIMESTAMP. Default to TRUE (use solarTIMESTAMP).
- **probs**: numeric vector of probabilities for quantile
- **tidy**: Logical indicating if the metrics must be returned in a tidy format (a long tibble, each observation in its own row)
- **metadata**: metadata object, usually the result of `read_sfn_metadata`. Only used if tidy is TRUE.
- **period**: Time period to aggregate data by. See period section for an explanation about the periods ('3 hours', '1 day', '1 month', '1 year', ...)
- **int_start**: Integer value indicating the starting hour of the special interval in 24h format. See Interval section in details.
- **int_end**: Integer value indicating the ending hour of the special interval in 24h format. See Interval section in details.
Details

*_metrics* functions are wrappers for *sfn_metrics* with a set of fixed arguments. *
_metrics* functions return all or some of the following statistics:

- **mean**: mean of variable (tree or environmental variable) for the given period. NAs are removed
- **sd**: standard deviation of the variable for the given period. NAs are removed
- **coverage**: Data coverage percentage (percentage of measures without NAs)
- **q_XX**: 0.XX quantile value for the period
- **centroid**: Diurnal centroid value (hours passed until the half of the summed daily value was reached). Only returned for sapflow measures when period is '1 day'
- **accumulated**: Accumulated values for precipitation only

Value

If tidy is TRUE, a tibble with the metrics for sapflow and environmental data, with all the metadata included. If tidy is FALSE (default), a list of tibbles with the calculated metrics.

daily_metrics

daily_metrics summarise daily data for all hours in the day

monthly_metrics

monthly_metrics summarise monthly data for all hours in the day.

nightly_metrics

**nightly_metrics** will return the metrics for night periods, summarised daily or monthly

Night for daily period starts in DOY x and ends in DOY x+1 (i.e. if night_start = 20, night_end = 6 values for the night starting at 2018-03-28 20:00:00 and ending at 2018-03-29 06:00:00 are summarised).

Night for monthly period summarises all night periods in the month, that includes from 00:00:00 of the first month night to 23:59:59 of the last month night.

daylight_metrics

daylight_metrics will return the metrics for daylight periods, summarised daily or monthly. Daylight interval is selected by start and end hours.

predawn_metrics

**predawn_metrics** will always return the metrics for predawn period, summarised daily or monthly. Predawn interval is selected by start and end hours.

Predawn metrics did not return the centroid metric.
midday_metrics

midday_metrics will always return the metrics for midday period, summarised daily or monthly. midday interval is selected by start and end hours.
Midday metrics did not return the centroid metric.

See Also

Other metrics: sfn_metrics()

Examples

```r
## daily_metrics
# data load
data('ARG_TRE', package = 'sapfluxnetr')
data('sfn_metadata_ex', package = 'sapfluxnetr')

# non tidy raw metrics (default)
ARG_TRE_raw_daily <- daily_metrics(ARG_TRE)
str(ARG_TRE_raw_daily)

# tidy daily metrics
ARG_TRE_daily <- daily_metrics(
  ARG_TRE, tidy = TRUE, metadata = sfn_metadata_ex
)
ARG_TRE_daily

## monthly_metrics
# data load
data('ARG_TRE', package = 'sapfluxnetr')
data('sfn_metadata_ex', package = 'sapfluxnetr')

# non tidy raw metrics (default)
ARG_TRE_raw_monthly <- monthly_metrics(ARG_TRE)
str(ARG_TRE_raw_monthly)

# tidy monthly metrics
ARG_TRE_monthly <- monthly_metrics(
  ARG_TRE, tidy = TRUE, metadata = sfn_metadata_ex
)
ARG_TRE_monthly

## nightly_metrics
# data load
data('AUS_CAN_ST2_MIX', package = 'sapfluxnetr')

# non tidy daily night metrics (default)
```
AUS_CAN_ST2_MIX_night <- nightly_metrics(AUS_CAN_ST2_MIX)

str(AUS_CAN_ST2_MIX_night)
AUS_CAN_ST2_MIX_night[['sapf']]
AUS_CAN_ST2_MIX_night[['env']]

# change the night interval
AUS_CAN_ST2_MIX_night_short <- nightly_metrics(
  AUS_CAN_ST2_MIX, int_start = 21, int_end = 4 # night starting and ending hour
)
AUS_CAN_ST2_MIX_night_short[['env']]  

# tidy nightly metrics
data('sfn_metadata_ex', package = 'sapfluxnetr')
AUS_CAN_ST2_MIX_night_tidy <- nightly_metrics(
  AUS_CAN_ST2_MIX,
  tidy = TRUE, metadata = sfn_metadata_ex
)
AUS_CAN_ST2_MIX_night_tidy

## daylight_metrics
# data load
data('AUS_CAN_ST2_MIX', package = 'sapfluxnetr')

# non tidy daily daylight metrics (default)
AUS_CAN_ST2_MIX_daylight <- daylight_metrics(AUS_CAN_ST2_MIX)

str(AUS_CAN_ST2_MIX_daylight)
AUS_CAN_ST2_MIX_daylight[['sapf']]  
AUS_CAN_ST2_MIX_daylight[['env']]  

# change the daylight interval
AUS_CAN_ST2_MIX_daylight_short <- daylight_metrics(
  AUS_CAN_ST2_MIX, int_start = 8, int_end = 18 # night starting and ending hour
)
AUS_CAN_ST2_MIX_daylight_short[['env']]  

# tidy daylight metrics
data('sfn_metadata_ex', package = 'sapfluxnetr')
AUS_CAN_ST2_MIX_daylight_tidy <- daylight_metrics(
  AUS_CAN_ST2_MIX,
  tidy = TRUE, metadata = sfn_metadata_ex
)
AUS_CAN_ST2_MIX_daylight_tidy

## predawn_metrics
# data load
data('AUS_CAN_ST2_MIX', package = 'sapfluxnetr')
# non tidy daily predawn metrics (default)
AUS_CAN_ST2_MIX_predawn <- predawn_metrics(AUS_CAN_ST2_MIX)

str(AUS_CAN_ST2_MIX_predawn)
AUS_CAN_ST2_MIX_predawn[['sapf']]
AUS_CAN_ST2_MIX_predawn[['env']]

# change the predawn interval
AUS_CAN_ST2_MIX_predawn_short <- predawn_metrics(AUS_CAN_ST2_MIX, int_start = 8, int_end = 18 # night starting and ending hour)
AUS_CAN_ST2_MIX_predawn_short[['env']]

# tidy daylight metrics
data('sfn_metadata_ex', package = 'sapfluxnetr')
AUS_CAN_ST2_MIX_predawn_tidy <- predawn_metrics(AUS_CAN_ST2_MIX, tidy = TRUE, metadata = sfn_metadata_ex)
AUS_CAN_ST2_MIX_predawn_tidy

## midday_metrics
# data load
data('AUS_CAN_ST2_MIX', package = 'sapfluxnetr')

# non tidy daily midday metrics (default)
AUS_CAN_ST2_MIX_midday <- midday_metrics(AUS_CAN_ST2_MIX)

str(AUS_CAN_ST2_MIX_midday)
AUS_CAN_ST2_MIX_midday[['sapf']]
AUS_CAN_ST2_MIX_midday[['env']]

# change the midday interval
AUS_CAN_ST2_MIX_midday_short <- midday_metrics(AUS_CAN_ST2_MIX, int_start = 8, int_end = 18 # night starting and ending hour)
AUS_CAN_ST2_MIX_midday_short[['env']]

# tidy daylight metrics
data('sfn_metadata_ex', package = 'sapfluxnetr')
AUS_CAN_ST2_MIX_midday_tidy <- midday_metrics(AUS_CAN_ST2_MIX, tidy = TRUE, metadata = sfn_metadata_ex)
AUS_CAN_ST2_MIX_midday_tidy
metrics_tidyfier

Build a tidy data frame from the metrics results nested list

Description

Transform the nested list of metrics in a tidy tibble where each observation has its own row

Usage

metrics_tidyfier(
  metrics_res,
  metadata,
  interval = c("general", "predawn", "midday", "night", "daylight")
)

Arguments

- **metrics_res**: Nested list containing the metrics results as obtained from `metrics`
- **metadata**: List containing the metadata nested list, as obtained from `read_sfn_metadata`
- **interval**: Interval to return, it depends on the `metrics_res` and can be "gen" for the general metrics, "md" for midday metrics, "pd" for predawn metrics, "night" for night metrics or "day" for diurnal metrics.

Value

a tibble with the following columns:

- **TIMESTAMP**: POSIXct vector with the date-time of the observation
- **si_code**: Character vector with the site codes
- **pl_code**: Character vector with the plant codes
- **sapflow_***: Variables containing the different metrics for the sapflow measurements (i.e. sapflow_mean, sapflow_q_95)
- **ta_*; rh_*; vpd_*; ...**: Variables containing the different metrics for environmental variables (i.e. ta_mean, ta_q_95)
- **pl_***: plant metadata variables (i.e. pl_sapw_area, pl_sens_meth)
- **si_***: site metadata variables (i.e. si_biome, si_contact_firstname)
- **st_***: stand metadata variables (i.e. st_aspect, st_lai)
- **sp_***: species metadata variables (i.e. sp_basal_area_perc)
- **env_***: environmental metadata variables (i.e. env_timezone)
Examples

```r
# data
multi_sfn <- sfn_data_multi(ARG_TRE, ARG_MAZ, AUS_CAN_ST2_MIX)
data('sfn_metadata_ex', package = 'sapfluxnetr')

# metrics
multi_metrics <- daily_metrics(multi_sfn)

# tidyfing
multi_tidy <- metrics_tidyfier(
  multi_metrics, sfn_metadata_ex, interval = 'general'
)
multi_tidy

# A really easier way of doing the same
multi_tidy_easy <- daily_metrics(multi_sfn, tidy = TRUE, metadata = sfn_metadata_ex)
```

---

**norm_diurnal_centroid**  
*Normalized diurnal centroid calculation*

**Description**

Calculate the normalized diurnal centroid for sapflow variables

**Usage**

```r
norm_diurnal_centroid(sapf_var, rad_var)
```

**Arguments**

- **sapf_var**: A numeric vector containing the sapflow values for a day at a regular intervals. Missing values are allowed but not recommended.
- **rad_var**: A numeric vector containing the incoming radiation for a day at a regular intervals. Missing values are allowed but not recommended. Must be the same length as sapf_var.

**Details**

The code for this function has been kindly provided by Jacob Nelson in python (see [https://github.com/jnelson18/FluxnetTools/blob/master/FileS3.py](https://github.com/jnelson18/FluxnetTools/blob/master/FileS3.py)) and has been translated to a tidy data philosophy in R to be used inside a `summarise` statement.

**Value**

A numeric value with the normalized diurnal centroid value
Normalized diurnal centroid algorithm

This function calculates the diurnal centroid of sapflow measures relative to the diurnal centroid of incoming radiation (in any units). For that the incoming radiation diurnal centroid is substracted from the sapflow diurnal centroid:

\[ Sapf_{cent} - \text{IncomingRad}_{cent} \]

Author(s)

Jacob Nelson & Víctor Granda

---

**Description**

Given a site code and a route, `read_sfn_data` will return the selected sfn_data object

**Usage**

```r
read_sfn_data(site_codes, folder = ".")
```

**Arguments**

- `site_codes`: A character vector with the site code/s
- `folder`: Route to the folder containing the .RData file. Default to working directory.

**Value**

If `site_codes` is a vector of length 1, an sfn_data object with the selected site data. If `site_codes` is a vector of length > 1, then a sfn_data_multi object containing all selected sites.

**Examples**

```r
# Let's access the data in "folder". This typically is the folder where the sapflow data at the desired unit level is (i.e. "RData/plant"), but in this example we will create a temporal folder with some data to test the function
folder <- tempdir()
save(ARG_TRE, file = file.path(folder, 'ARG_TRE.RData'))
save(ARG_MAZ, file = file.path(folder, 'ARG_MAZ.RData'))

# now we read a single site
ARG_TRE_test <- read_sfn_data('ARG_TRE', folder)
ARG_TRE_test

# or we can read multiple sites at once
multi_sfn <- read_sfn_data(
  c('ARG_TRE', 'ARG_MAZ'), folder
)```
read_sfn_metadata

Description

Read metadata from all sites in folder and write it to disk to cache the info for easy and fast access.

Usage

read_sfn_metadata(folder = ".", .write_cache = FALSE)

Arguments

folder Route to the folder containing the data. Default to working directory
.write_cache Logical indicating if a cached copy of the metadata must be written in folder.

Details

Load all data in memory to collect metadata info can be resource limiting. For easy and quick access to metadata, this function stores an .RData file in the specified folder along the data with all the metadata preloaded. Also it return it as an object to use in filtering and selecting sites.

Value

A list of tibbles with the five metadata classes (site, stand, species, plant and environmental)

Examples

# Let's access the data in "folder". This typically is the folder where the
# sapflow data at the desired unit level is (i.e. "RData/plant"), but in this
# example we will create a temporal folder with some data to test the function
c tempdir()
save(ARG_TRE, file = file.path(folder, 'ARG_TRE.RData'))
save(ARG_MAZ, file = file.path(folder, 'ARG_MAZ.RData'))

# create and load the metadata. The first time we use .write_cache = TRUE,
# to ensure creating a file containing the metadata for speed the process
# for the next times
read_sfn_metadata(
  folder = folder, .write_cache = TRUE
)
# a cached copy must have been written to "folder"
file.exists(paste0(folder, '.metadata_cache.RData')) # TRUE

# after that, we only need to espcify the folder
sites_metadata <- read_sfn_metadata(folder = folder)  # quicker than before
sites_metadata

sfn_data-class  
S4 class for sapfluxnet site data

Description
Main class for storing sapfluxnet project site data and metadata

Details
This class allows to store all the data and metadata for a sapfluxnet site in one single object, to easily work with it. See vignette('sfn-data-classes', package = 'sapfluxnetr') for more info.

Slots

  sapf_data  A data frame with the sapf data
  env_data  A data frame with the env data
  sapf_flags  A data frame with the same dimensions of sapf_data with the flag info for each tree/TIMESTAMP combination
  env_flags  A data frame with the same dimensions of env_data with the flag info for each env_var/TIMESTAMP combination
  si_code  A character vector of length one indicating the site code
  timestamp  A POSIXct vector of length nrow(sapf_data) with the timestamp
  solar_timestamp  A POSIXct vector of length nrow(sapf_data) with the solar timestamp
  site_md  A data frame containing the site metadata
  stand_md  A data frame containing the stand metadata
  species_md  A data frame containing the species metadata
  plant_md  A data frame containing the plant metadata
  env_md  A data frame containing the env metadata

sfn_data_multi-class  
S4 class for sapfluxnet multi-site data

Description
Multi sfn data class, derived from list

Details
This class inherits from list, but modified to contain sfn_data objects as elements. This will allow to work with several sites at the same time obtaining results for all of them combined or individually as elements of the resulting list (with lapply or purrr::map)
sfn_data_multi_validity

Validity method for sfn_data_multi class

Description

Validation checks for generating sfn_data_multi class objects

Details

This method is used internally to ensure the correctness of the sfn_data_multi object. Basically ensures that the object returned is a list of sfn_data class objects

sfn_data_validity  
Validity method for sfn_data class

Description

Validation checks for generating sfn_data class objects

Details

This method is used internally when creating and/or modifying sfn_data class objects to ensure that the object returned is correct in terms of content classes and dimensions (i.e. sapflow data and environmental data has the same length)

sfn_filter  
Filter sfn_data by variable/s value

Description

Port of filter for sfn_data and sfn_data_multi objects

Usage

sfn_filter(sfn_data, ... , solar = FALSE)

Arguments

sfn_data  
sfn_data or sfn_data_multi object to subset

...  
expressions to pass to the filter function

solar  
Logical indicating if solar timestamp must used to subset
Details

'sfn_filter' will remove the rows not matching the logical expression/s provided. So, it will remove cases and will create TIMESTAMP gaps, so its use is not recommended except in the case of filtering by TIMESTAMP (i.e. to set several sites (sfn_data_multi) in the same time frame). For other scenarios (removing extreme environmental conditions values or strange sapflow measures patterns) see sfn_mutate and sfn_mutate_at.

Value

For sfn_data objects, a filtered sfn_data or NULL if no data meet the criteria. For sfn_data_multi another sfn_data_multi with the sites filtered, and an empty sfn_data_multi if any sites met the criteria.

Examples

```r
library(dplyr)
library(lubridate)

# data
data('ARG_TRE', package = 'sapfluxnetr')

# by timestamp
foo_timestamp <- get_timestamp(ARG_TRE)
foo_timestamp_trimmed <- foo_timestamp[1:100]
sfn_filter(
  ARG_TRE,
  TIMESTAMP %in% foo_timestamp_trimmed
)

# by wind speed value
ws_threshold <- 25
sfn_filter(
  ARG_TRE,
  ws <= ws_threshold
)

## multi
data('ARG_MAZ', package = 'sapfluxnetr')
multi_sfn <- sfn_data_multi(ARG_TRE, ARG_MAZ)

# by timestamp
sfn_filter(
  multi_sfn,
  between(day(TIMESTAMP), 18, 22)
)

# by wind speed value
sfn_filter(
```
sfn_get_generics

\[
\text{multi}_\text{sfn}, \\
\text{ws} \leq \text{ws}_\text{threshold}
\]

\[
\text{sfn_get}_{\text{generics}} \quad \text{sfn\_data custom get generics}
\]

**Description**

Generics for getting the info in the slots of SfnData

**Usage**

- `get_sapf_data(object, ...)`
- `get_env_data(object, ...)`
- `get_sapf_flags(object, ...)`
- `get_env_flags(object, ...)`
- `get_timestamp(object, ...)`
- `get_solar_timestamp(object, ...)`
- `get_si_code(object, ...)`
- `get_site_md(object, ...)`
- `get_stand_md(object, ...)`
- `get_species_md(object, ...)`
- `get_plant_md(object, ...)`
- `get_env_md(object, ...)`

**Arguments**

- `object`: Object to get data from
- `...`: Further arguments to pass to the corresponding get method

**Details**

see `sfn_get_methods` for detailed info about using the get methods in `sfn_data` class objects and `sfn_multi_get_methods` for detailed info about using the get methods in `sfn_data_multi` class objects.
sfn_get_methods

sfn_data get methods

Description
Methods to get the data and metadata from the sfn_data class slots

Usage

## S4 method for signature 'sfn_data'
get_sapf_data(object, solar = FALSE)

## S4 method for signature 'sfn_data'
get_env_data(object, solar = FALSE)

## S4 method for signature 'sfn_data'
get_sapf_flags(object, solar = FALSE)

## S4 method for signature 'sfn_data'
get_env_flags(object, solar = FALSE)

## S4 method for signature 'sfn_data'
get_timestamp(object)

## S4 method for signature 'sfn_data'
get_solar_timestamp(object)

## S4 method for signature 'sfn_data'
get_si_code(object)

## S4 method for signature 'sfn_data'
get_site_md(object)

## S4 method for signature 'sfn_data'
get_stand_md(object)

## S4 method for signature 'sfn_data'
get_species_md(object)

## S4 method for signature 'sfn_data'
get_plant_md(object)

## S4 method for signature 'sfn_data'
get_env_md(object)

Arguments

object Object of class sfn_data from which data is retrieved
solar Logical indicating if the timestamp to return in the get_sapf_data, get_env_data, get_sapf_flags and get_env_flags methods is the solarTIMESTAMP (TRUE) or the contributors provided TIMESTAMP (FALSE)

Details

get_sapf_data and get_env_data methods retrieve sapflow or environmental tibbles to create a functional dataset to work with.

get_sapf_flags and get_env_flags methods retrieve sapflow or environmental flags as tibbles.

get_timestamp and get_solar_timestamp methods retrieve only the timestamp as POSIXct vector.

get_si_code method retrieve a character vector with length(timestamp) containing the site code.

get_site_md, get_stand_md, get_species_md, get_plant_md and get_env_md methods retrieve the corresponding metadata.

Examples

library(dplyr)

data('ARG_TRE', package = 'sapfluxnetr')
sapf_data <- get_sapf_data(ARG_TRE, solar = TRUE)
env_data_no_solar <- get_env_data(ARG_TRE, solar = FALSE)
plant_md <- get_plant_md(ARG_TRE)

# dplyr pipe to get the mean dbh for a site
ARG_TRE %>%
  get_plant_md() %>%
  summarise(dbh_mean = mean(pl_dbh, na.rm = TRUE)) %>%
  pull(dbh_mean)

sfn_metadata_ex

sfn_metadata cache file for example data (ARG_MAZ, ARG_TRE and AUS_CAN_ST2_MIX)

Description

Example metadata cache file content for package usage demonstration

Usage

sfn_metadata_ex

Format

A list with five elements, each of one a metadata type.
Examples

data('sfn_metadata_ex', package = 'sapfluxnetr')
sfn_metadata_ex

---

## sfn_metrics

### Metrics summary function

#### Description

Generate metrics from a site/s data for the period indicated

#### Usage

```r
sfn_metrics(
  sfn_data,
  period,
  .funs,
  solar,
  interval = c("general", "predawn", "midday", "night", "daylight"),
  int_start = NULL,
  int_end = NULL,
...
)
```

#### Arguments

- **sfn_data**: `sfn_data` or `sfn_data_multi` object to obtain the metrics from
- **period**: Time period to aggregate data by. See period section for an explanation about the periods ('3 hours', '1 day', '1 month', '1 year', ...)
- **.funs**: List of function calls to summarise the data by, see .funs section for more details.
- **solar**: Logical indicating if the solarTIMESTAMP must be used instead of the site local TIMESTAMP. Default to TRUE (use solarTIMESTAMP).
- **interval**: Character vector indicating if the metrics must be filtered by an special hour interval. See Interval section in details.
- **int_start**: Integer value indicating the starting hour of the special interval in 24h format. See Interval section in details.
- **int_end**: Integer value indicating the ending hour of the special interval in 24h format. See Interval section in details.
- **...**: optional arguments to pass to methods used (i.e. .collapse_timestamp or summarise funs extra arguments)
Value

For `sfn_data` objects, a list of tbl_df objects with the following structure:

- `$sapf`: metrics for the sapflow data
- `$env`: metrics for the environmental data

For `sfn_data_multi` objects, a list of lists of tbl_df objects with the metrics for each site:

- `$SITE_CODE`
  - `$sapf`: metrics for the sapflow data
  - `$env`: metrics for the environmental data
- `$NEXT_SITE_CODE...`

Period

The `period` argument is used by the internal function `.collapse_timestamp` and it can be stated in two ways:

- `frequency period` format: "3 hours", "1 day", "7 days", "1 month"
- As a `custom function`. This will be the name of a function, without quotes, that accepts as the first argument the timestamp to collapse. The result of the function must be a vector of collapsed TIMESTAMPs of the same length than the original TIMESTAMP which will be used to group by and summarise the data. Additional arguments to this function, if needed, can be passed in the `...` argument.

`.collapse_timestamp` also accepts the `side` argument to collapse by the starting timestamp or the ending timestamp of each group. This can be supplied in the `...` argument.

.funs

The `.funs` argument uses the same method as the `.funs` argument in the `summarise_all` function of the `dplyr` package. Basically it accepts a list of function calls generated by `list()`. If you want to pass on a custom function you can specify it here. See details in `summarise_by_period` for more complex summarising functions declaration.

Interval

Previously to the metrics summary, data can be filtered by a special interval (i.e. predawn or nightly). This filtering can be specified with the `interval` argument as this:

- "general" (default). No special interval is used, and metrics are performed with all the data.
- "predawn". Data is filtered for predawn interval. In this case `int_start` and `int_end` must be specified as 24h value
- "midday". Data is filtered for midday interval. In this case `int_start` and `int_end` must be specified as 24h value
- "night". Data is filtered for night interval. In this case `int_start` and `int_end` must be specified as 24h value
- "daylight". Data is filtered for daylight interval. In this case `int_start` and `int_end` must be specified as 24h value
See Also

Other metrics: metrics

Examples

library(dplyr)

### general metrics
## sfn_data

data(’ARG_TRE’, package = ’sapfluxnetr’)
ARG_TRE_metrics <- sfn_metrics(
  ARG_TRE,
  period = ’7 days’,
  .funs = list(~ mean(., na.rm = TRUE), ~ sd(., na.rm = TRUE), ~ n()),
  solar = FALSE,
  interval = ’general’
)

str(ARG_TRE_metrics)
ARG_TRE_metrics[['sapf']]
ARG_TRE_metrics[['env']]

### sfn_data_multi

data(’ARG_MAZ’, package = ’sapfluxnetr’)
data(’AUS_CAN_ST2_MIX’, package = ’sapfluxnetr’)
multi_sfn <- sfn_data_multi(ARG_TRE, ARG_MAZ, AUS_CAN_ST2_MIX)

multi_metrics <- sfn_metrics(
  multi_sfn,
  period = ’7 days’,
  .funs = list(~ mean(., na.rm = TRUE), ~ sd(., na.rm = TRUE), ~ n()),
  solar = FALSE,
  interval = ’general’
)

str(multi_metrics)
multi_metrics[['ARG_TRE']][['sapf']]  

### midday metrics
ARG_TRE_midday <- sfn_metrics(
  ARG_TRE,
  period = ’1 day’,
  .funs = list(~ mean(., na.rm = TRUE), ~ sd(., na.rm = TRUE), ~ n()),
  solar = TRUE,
  interval = ’midday’, int_start = 11, int_end = 13
)

str(ARG_TRE_midday)
ARG_TRE_midday[['sapf']]

Description

Methods to get the data and metadata from the sfn_data class slots

Usage

```r
## S4 method for signature 'sfn_data_multi'
get_sapf_data(object, solar = FALSE)

## S4 method for signature 'sfn_data_multi'
get_env_data(object, solar = FALSE)

## S4 method for signature 'sfn_data_multi'
get_sapf_flags(object, solar = FALSE)

## S4 method for signature 'sfn_data_multi'
get_env_flags(object, solar = FALSE)

## S4 method for signature 'sfn_data_multi'
get_timestamp(object)

## S4 method for signature 'sfn_data_multi'
get_solar_timestamp(object)

## S4 method for signature 'sfn_data_multi'
get_si_code(object)

## S4 method for signature 'sfn_data_multi'
get_site_md(object, collapse = FALSE)

## S4 method for signature 'sfn_data_multi'
get_stand_md(object, collapse = FALSE)

## S4 method for signature 'sfn_data_multi'
get_species_md(object, collapse = FALSE)

## S4 method for signature 'sfn_data_multi'
get_plant_md(object, collapse = FALSE)

## S4 method for signature 'sfn_data_multi'
get_env_md(object, collapse = FALSE)
```
Arguments

- **object**: Object of class `sfn_data_multi` from which data or metadata is retrieved.
- **solar**: Logical indicating if the timestamp to return in the `get_sapf_data`, `get_env_data`, `get_sapf_flags` and `get_env_flags` methods is the solar TIMESTAMP (TRUE) or the contributors provided TIMESTAMP (FALSE).
- **collapse**: Logical indicating if the metadata get methods must collapse the returning list to a data frame with all sites.

Details

- `get_sapf_data` and `get_env_data` methods retrieve sapflow or environmental tibbles from the `sfn_data` objects contained in the `sfn_data_multi` and return them in a list.
- `get_sapf_flags` and `get_env_flags` methods retrieve sapflow or environmental flags tibbles from the `sfn_data` objects contained in the `sfn_data_multi` and return them in a list.
- `get_timestamp` and `get_solar_timestamp` methods retrieve only the timestamps as POSIXct vectors and return them as a list (each element corresponding to a site in the `sfn_data_multi` object).
- `get_si_code` method retrieve a character vector with length(timestamp) containing the site code for each site, returning them as a list.
- `get_site_md`, `get_stand_md`, `get_species_md`, `get_plant_md` and `get_env_md` methods retrieve the corresponding metadata objects for each site returning them as a list, unless collapse is TRUE, then the list collapses to a tibble.

Examples

```r
library(dplyr)
```

---

**sfn_mutate**

*Mutate variables by function*

**Description**

Port of `mutate` for `sfn_data` and `sfn_data_multi` objects.

**Usage**

`sfn_mutate(sfn_data, ..., solar = FALSE)`

**Arguments**

- **sfn_data**: `sfn_data` or `sfn_data_multi` object to subset.
- **...**: Name-value pairs of expressions to pass to the `mutate` function.
- **solar**: Logical indicating if solar timestamp must used to subset.
Details

‘sfn_mutate‘ function will maintain the same number of rows before and after the modification, so it is well suited to modify variables without creating TIMESTAMP gaps (i.e. to change variable units). For mutating groups of variables at the same time see sfn_mutate_at.

Value

For sfn_data objects, a mutated sfn_data. For sfn_data_multi another sfn_data_multi with the sites mutated

Sapflow and environmental variables

‘sfn_mutate‘ internally joins the sapflow and environmental datasets by the TIMESTAMP, so it is possible to mutate variables conditionally between sapflow and environmental measures (i.e. mutate sapflow when wind is high or radiation is zero). Due to this, at the moment any new variable is dropped when building the final results, so this is ONLY intended to mutate existing variables without changing the names.

Examples

```r
library(dplyr)
library(lubridate)

# data
data('ARG_TRE', package = 'sapfluxnetr')

# transform to NAs any wind value above 25
ws_threshold <- 25
sfn_mutate(ARG_TRE, ws = if_else(ws > 25, NA_real_, ws))

## multi
data(ARG_MAZ, package = 'sapfluxnetr')
data(AUS_CAN_ST2_MIX, package = 'sapfluxnetr')
multi_sfn <- sfn_data_multi(ARG_TRE, ARG_MAZ, AUS_CAN_ST2_MIX)

multi_sfn_mutated <- sfn_mutate(
    multi_sfn, ws = if_else(ws > 25, NA_real_, ws)
)

multi_sfn_mutated[['ARG_TRE']]`
**Usage**

sfn_mutate_at(sfn_data, .vars, .funs, ..., solar = FALSE)

**Arguments**

- **sfn_data**: sfn_data or sfn_data_multi object to subset
- **.vars**: Variables to mutate. Passed to `mutate_at`
- **.funs**: Function/s for mutate, passed to `mutate_at`
- **...**: Extra arguments to pass to the functions in .funs, passed to `mutate_at`
- **solar**: Logical indicating if solar timestamp must used to subset

**Details**

'sfn_mutate_at' function will maintain the same number of rows before and after the modification, so it is well suited to modify variables without creating TIMESTAMP gaps (i.e. to change variable units). For mutating individual variables see `sfn_mutate`.

**Value**

For `sfn_data` objects, a mutated `sfn_data`. For `sfn_data_multi` another `sfn_data_multi` with the sites mutated

**Examples**

```r
library(dplyr)
library(lubridate)

# data
data('ARG_TRE', package = 'sapfluxnetr')

# transform to NAs any sapflow value occured with wind speed above 25
ws_threshold <- 25
# get the names of the variables to mutate (tree names)
vars_to_mutate <- names(get_sapf_data(ARG_TRE)[,-1]) # no TIMESTAMP
sfn_mutate_at(
  ARG_TRE,
  .vars = vars(one_of(vars_to_mutate)),
  .funs = list(
    ~ case_when(
      ws > ws_threshold ~ NA_real_,
      TRUE ~ .
    )
  )
)

## multi
data(ARG_MAZ, package = 'sapfluxnetr')
data(AUS_CAN_ST2_MIX, package = 'sapfluxnetr')
multi_sfn <- sfn_data_multi(ARG_TRE, ARG_MAZ, AUS_CAN_ST2_MIX)
```
## in multi it’s better to discard the variables to not mutate:
vars_to_not_mutate <- names(get_env_data(ARG_TRE)) # all the environmental

multi_sfn_mutated <- sfn_mutate_at(
  multi_sfn,
  .vars = vars(-one_of(vars_to_not_mutate)), # we use -
  .funs = list(
    - case_when(
      ws > ws_threshold ~ NA_real_,
      TRUE ~ .
    )
  )
)

multi_sfn_mutated[['ARG_TRE']]
Value

A ggplot object that can be called to see the plot. If input is an sfn_data_multi object, a list with the plots.

ggplot plotting system

plot is a base R function which uses the base R plotting system to show the plot. We prefer the ggplot plotting system, which allow for storing the plots in objects and can be subjected to further modifications. This allow the package users to generate rather simple plots that can be fine tuned afterwards to the user taste. Generating a plot method for the sfn_data class returning a ggplot object is not desired (it change the way plot works and can be misleading about the plot general usage). So, instead, we offer this function, sfn_plot.

Type

The type argument controls what is going to be plotted. It accepts the following:

- "sapf": It will plot sapflow data vs. TIMESTAMP
- "env": It will plot environmental variables vs. TIMESTAMP
- "ta", "rh", "vpd", "ppfd_in", "netrad", "sw_in", "ext_rad", "ws", "precip", "swc_shallow" and "swc_deep": They will plot the corresponding variable vs. TIMESTAMP

Formula

The formula argument can be used to select an environmental variable to plot versus all the sapflow measurements. Any environmental variable is allowed, if it exist in the site provided.

Geometry

By default sfn_plot generates plots using geom_point geometry, except in the case of type = "ws" and type = "precip" where geom_col is used. These geometries can be modified with the ... argument.

Examples

library(ggplot2)

# data
data('ARG_TRE', package = 'sapfluxnetr')

# plotting directly
sfn_plot(ARG_TRE, type = 'sapf')

# this could be noisy, you can facet by "Tree" (for sapflow) or by
# "Variable" (for environmental data):
sfn_plot(ARG_TRE, type = 'sapf') +
  facet_wrap(~ Tree)

sfn_plot(ARG_TRE, type = 'env') +
  facet_wrap(~ Variable, scales = 'free_y')
# saving and modifying:
env_plot <- sfn_plot(ARG_TRE, type = 'env', solar = FALSE) +
  facet_wrap(~ Variable, scales = 'free_y')
env_plot + labs(title = 'Environmental variables facet plot')

# formula
sfn_plot(ARG_TRE, formula_env = ~ vpd)

---

## Description

Generic functions for replacement functions for sfn_data

## Usage

```r
get_sapf_data(object) <- value
get_env_data(object) <- value
get_sapf_flags(object) <- value
get_env_flags(object) <- value
get_timestamp(object) <- value
get_solar_timestamp(object) <- value
get_si_code(object) <- value
get_site_md(object) <- value
get_stand_md(object) <- value
get_species_md(object) <- value
get_plant_md(object) <- value
get_env_md(object) <- value
```

## Arguments

- **object**: Object to replace
- **value**: Object to replace with
Details

see `sfn_replacement_methods` for more info about using the replacement methods in `sfn_data` objects

Description

Methods to replace the data and metadata from the `sfn_data` class slots

Usage

```r
## S4 replacement method for signature 'sfn_data'
get_sapf_data(object) <- value

## S4 replacement method for signature 'sfn_data'
get_env_data(object) <- value

## S4 replacement method for signature 'sfn_data'
get_sapf_flags(object) <- value

## S4 replacement method for signature 'sfn_data'
get_env_flags(object) <- value

## S4 replacement method for signature 'sfn_data'
get_timestamp(object) <- value

## S4 replacement method for signature 'sfn_data'
get_solar_timestamp(object) <- value

## S4 replacement method for signature 'sfn_data'
get_si_code(object) <- value

## S4 replacement method for signature 'sfn_data'
get_site_md(object) <- value

## S4 replacement method for signature 'sfn_data'
get_stand_md(object) <- value

## S4 replacement method for signature 'sfn_data'
get_species_md(object) <- value

## S4 replacement method for signature 'sfn_data'
get_plant_md(object) <- value
```
## S4 replacement method for signature 'sfn_data'

```r
get_env_md(object) <- value
```

### Arguments

- **object**: sfn_data containing the slot to replace
- **value**: object with the data to replace sfn_data slot with

### Details

The replacement object must be a valid object for that slot:

- For `get_sapf_data`, `get_env_data`, `get_sapf_flags` and `get_env_flags` a data.frame or tibble without the `TIMESTAMP` variable
- For `get_*_md` a data.frame or tibble
- For `get_timestamp` and `get_solar_timestamp` a POSIXct vector of length == nrow(sapf/env_data)
- For `get_si_code` a character vector

Validity is automatically checked before modifying the sfn_data object, and an error is raised if not valid

### Examples

```r
# preparation
data('ARG_TRE', package = 'sapfluxnetr')
sapf_data <- get_sapf_data(ARG_TRE, solar = TRUE)

# modifying the slot data
sapf_data[1:10, 2] <- NA

# replacement. Remember, the sfn_data slot does not contain a TIMESTAMP
# variable, it must be removed
get_sapf_data(ARG_TRE) <- sapf_data[, -1]
```

---

### sfn_sites_in_folder

**list available sites in a db folder**

### Description

Retrieves the site codes in the specified folder

### Usage

```r
sfn_sites_in_folder(folder = ".")
```
sfn_vars_to_filter

Arguments
folder Character vector of length 1 indicating the route to the db folder

Details
If folder

Value
A character vector with the site codes present in the folder, an error if the folder is not valid or does not contain any site data file.

Examples
# Let's access the data in "folder". This typically is the folder where the sapflow data at the desired unit level is (i.e. "RData/plant"), but in this example we will create a temporal folder with some data to test the function
folder <- tempdir()
save(ARG_TRE, file = file.path(folder, 'ARG_TRE.RData'))
save(ARG_MAZ, file = file.path(folder, 'ARG_MAZ.RData'))
save(AUS_CAN_ST2_MIX, file = file.path(folder, 'AUS_EUC_ST2_MIX.RData'))

# lets see the sites
sites <- sfn_sites_in_folder(folder)

sfn_vars_to_filter List all variables that can be used to filter sites

Description
sfn_vars_to_filter() returns a list with the variables for each kind of metadata that can be used to select and filter sites

Usage
sfn_vars_to_filter()

Value
A list with five elements, site_md, stand_md, species_md, plant_md and env_md

Examples
# all variables
sfn_vars_to_filter()

# by some metadata
sfn_vars_to_filter()$site_md
### show.sfn_data-method

**Show method for sfn_data**

**Description**

print a summary for `sfn_data` objects

**Usage**

```r
## S4 method for signature 'sfn_data'
show(object)
```

**Arguments**

- `object` sfn_data object to show

### show.sfn_data_multi-method

**Show method for sfn_data_multi**

**Description**

print a summary for `sfn_data_multi` objects

**Usage**

```r
## S4 method for signature 'sfn_data_multi'
show(object)
```

**Arguments**

- `object` sfn_data_multi object to show
**summarise_by_period**

**Summaries by period**

**Description**

This function collapse the TIMESTAMP to the desired period (day, month...) by setting the same value to all timestamps within the period. This modified TIMESTAMP is used to group by and summarise the data.

**Usage**

```r
summarise_by_period(data, period, .funs, ...)
```

**Arguments**

- `data` sapflow or environmental data as obtained by `get_sapf_data` and `get_env_data`. Must have a column named TIMESTAMP.
- `period` period to collapse by. See `sfn_metrics` for details.
- `.funs` funs to summarise the data. See details.
- `...` optional arguments. See details.

**Details**

This function uses internally `.collapse_timestamp` and `summarise_all`. Arguments to control these functions can be passed as `...`. Arguments for each function are spliced and applied when needed. Be advised that all arguments passed to the summarise_all function will be applied to all the summarising functions used, so it will fail if any of that functions does not accept that argument. To complex function-argument relationships, indicate each summary function call within the `.funs` argument as explained here `summarise_all`:

```r
# This will fail because na.rm argument will be also passed to the n function,
# which does not accept any argument:
summarise_by_period(
  data = get_sapf_data(ARG_TRE),
  period = '7 days',
  .funs = list(mean, sd, n()),
  na.rm = TRUE
)

# to solve this is better to use the .funs argument:
summarise_by_period(
  data = get_sapf_data(ARG_TRE),
  period = '7 days',
  .funs = list(~ mean(.), na.rm = TRUE), ~ sd(.), na.rm = TRUE), ~ n())
```
Value

A ‘tbl_df’ object with the metrics results. The names of the columns indicate the original variable (tree or environmental variable) and the metric calculated (i.e. ‘vpd_mean’), separated by underscore.

TIMESTAMP_coll

Previously to the collapsing step, a temporal variable called TIMESTAMP_coll is created to be able to catch the real timestamp when some events happens, for example to use the min_time function. If your custom summarise function needs to get the time at which some event happens, use TIMESTAMP_coll instead of TIMESTAMP for that:

```r
min_time <- function(x, time) {
  time[which.min(x)]
}
summarise_by_period(
  data = get_sapf_data(ARG_TRE),
  period = '1 day',
  .funs = list(~ min_time(. , time = TIMESTAMP_coll)) # Not TIMESTAMP
)
```

Examples

```r
library(dplyr)
# data
data('ARG_TRE', package = 'sapfluxnetr')

# simple summary
summarise_by_period(
  data = get_sapf_data(ARG_TRE),
  period = '7 days',
  .funs = list(~ mean(. , na.rm = TRUE), ~ sd(. , na.rm = TRUE), ~ n())
)
```

Reexporting the pipe operator

Description

Imported from magrittr package
Examples

# piping sites
ARG_TRE %>% daily_metrics()
Index

* datasets
  ARG_MAIZ, 3
  ARG_TRE, 3
  AUS_CAN_ST2_MIX, 4
  sfn_metadata_ex, 26

* metrics
  metrics, 11
  sfn_metrics, 27

.collapse_timestamp, 12, 28, 41
%>%, 42

ARG_MAIZ, 3
ARG_TRE, 3
AUS_CAN_ST2_MIX, 4

ceiling_date, 12

daily_metrics (metrics), 11
data_coverage, 4
daylight_metrics (metrics), 11
describe_md_variable, 5
diurnal_centroid, 6

filter, 7, 22
filter_sites_by_md, 7
floor_date, 12

geom_col, 34, 35
geom_point, 34, 35
get_env_data, 41
get_env_data (sfn_get_generics), 24
get_env_data, sfn_data-method
(sfn_get_methods), 25
get_env_data, sfn_data_multi-method
(sfn_multi_get_methods), 30
get_env_data<-
(sfn_replacement_generics), 36
get_env_data<-, sfn_data-method
(sfn_replacement_methods), 37
get_env_md (sfn_get_generics), 24
get_env_md, sfn_data-method
(sfn_get_methods), 25
g
sfn_multi_get_methods, 24, 30
sfn_mutate, 23, 31, 33
sfn_mutate_at, 23, 32, 32
sfn_plot, 34
sfn_replacement_generics, 36
sfn_replacement_methods, 37, 37
sfn_sites_in_folder, 7, 38
sfn_vars_to_filter, 39
show, sfn_data-method, 40
show, sfn_data_multi-method, 40
summarise, 6, 18
summarise_all, 28, 41
summarise_by_period, 28, 41