Package 'SWMPr'

November 10, 2014

Type Package

Title SWMPr package for estuarine monitoring data

Version 0.4.0

Date 2014-11-04

Author Marcus Beck

Maintainer Marcus Beck <mbafs2012@gmail.com>

Description This packages provides functions for retrieving, organizing, and analyzing monitoring data from the National Estuarine Research Reserve System. Data that can be used with this package are collected as part of the System Wide Monitoring Program and maintained online at the Centralized Data Management Office.

BugReports https://github.com/fawda123/SWMPr/issues

License CC0

Imports data.table,plyr,reshape2,XML

Suggests SSOAP

Depends R (>= 3.0.0),ggmap,ggplot2,zoo

R topics documented:

`aggregate.swmpr` *Aggregate swmpr data*

Description

Aggregate swmpr data by specified time period and method

Usage

```
## S3 method for class 'swmpr'
aggregate(x, by, FUN = function(x) mean(x, na.rm = TRUE),
  params = NULL, aggs_out = FALSE, na.action = na.pass, ...)
```

Arguments

`x`	input swmpr object
`by`	chr string of time period for aggregation one of `'years'`, `'quarters'`, `'months'`, `'weeks'`, `'days'`, or `'hours'`
`FUN`	aggregation function, default `mean` with `na.rm = TRUE`
`params`	names of parameters to aggregate, default all
`aggs_out`	logical indicating if `data.frame` is returned of raw data with datetimestamp formatted as aggregation period, default `FALSE`
`na.action`	function for treating missing data, default `na.pass`
`...`	additional arguments passed to other methods

Details

The `aggregate` function summarizes or condenses parameter data for a swmpr object by set periods of observation and a user-supplied function. It is most useful for aggregating noisy data to evaluate trends on longer time scales, or to simply reduce the size of a dataset. Data can be aggregated by `'years'`, `'quarters'`, `'months'`, `'weeks'`, `'days'`, or `'hours'` for the supplied function, which defaults to the `mean`. A swmpr object is returned for the aggregated data, although the datetimestamp vector will be converted to a date object if the aggregation period is a day or longer. Days are assigned to the date vector if the aggregation period is a week or longer based on the round method for `IDate` objects. This approach was used to facilitate plotting using predefined methods for Date and POSIX objects.

The method of treating NA values for the user-supplied function should be noted since this may greatly affect the quantity of data that are returned (see the examples). Finally, the default argument for `na.action` is set to `na.pass` for swmpr objects to preserve the time series of the input data.

Value

Returns an aggregated swmpr object. QAQC columns are removed if included with input object.

See Also

aggregate

Examples

```
## get data, prep
path <- system.file('zip_ex', package = 'SWMPr')
dat <- import_local(path, 'apacpwq')
swmpr_in <- subset(qaqc(dat), rem_cols = TRUE)

## get mean DO by quarters
aggregate(swmpr_in, 'quarters', params = c('do_mgl'))

## get variance of DO by years, remove NA when calculating variance
## omit NA data in output
fun_in <- function(x)  var(x, na.rm = TRUE)
aggregate(swmpr_in, FUN = fun_in, 'years')
```

---

all_params *Import current station records from the CDMO*

Description

Import current station records from the CDMO starting with the most current date

Usage

```
all_params(station_code, Max = 100)
```

Arguments

station_code	chr string of station, 7 or 8 characters
Max	numeric value for number of records to obtain from the current date, maximum of 100

Details

This function uses the SSOAP package to retrieve data from the CDMO through a SOAP client interface. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: http://cdmo.baruch.sc.edu/webservices.cfm. Function is the CDMO equivalent of exportAllParamsXMLNew.

Value

Returns a swmpr object, all available parameters including QAQC columns

See Also

all_params_dtrng, single_param

Examples

```
## Not run:

## all parameters for a station, most recent
all_params('hudscwq')


## End(Not run)
```

`all_params_dtrng` *Get CDMO records within a date range*

Description

Get station records from the CDMO within a date range

Usage

```
all_params_dtrng(station_code, dtrng, param = NULL)
```

Arguments

`station_code`	chr string of station, 7 or 8 characters
`dtrng`	two element chr string, each of format MM/DD/YYYY
`param`	chr string for a single parameter to return, defaults to all parameters for a station type.

Details

This function uses the SSOAP package to retrieve data from the CDMO through a SOAP client interface. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: http://cdmo.baruch.sc.edu/webservices.cfm. This function is the CDMO equivalent of `exportAllParamsDateRangeXMLNew`.

Value

Returns a swmpr object, all parameters for a station type (nutrients, water quality, or meteorological) or a single parameter if specified. QAQC columns are not provided for single parameters. Up to 1000 records can be obtained.

See Also

`all_params`, `single_param`

Examples

```
## Not run:

## get all parameters within a date range
all_params_dtrng('hudscwq', c('09/10/2012', '02/8/2013'))

## get single parameter within a date range
all_params_dtrng('hudscwq', c('09/10/2012', '02/8/2013'),
   param = 'do_mgl')


## End(Not run)
```

---

comb *Combine swmpr data*

Description

Combine swmpr data types for a station by common time series

Usage

```
comb(...)

## S3 method for class 'swmpr'
comb(..., timestep = 15, differ = timestep/2,
  method = "union")
```

Arguments

`...`	swmpr object input from one to many
`timestep`	numeric value of time step to use in minutes, passed to `setstep`
`differ`	numeric value defining buffer for merging time stamps to standardized time series, passed to `setstep`
`method`	chr string indicating method of combining (`'union'` for all dates as continuous time series, `'intersect'` for areas of overlap, or `'station'` for a given station)

Details

The `comb` function is used to combine multiple swmpr objects into a single object with a continuous time series at a given step. The `timestep` function is used internally such that `timestep` and `differ` are accepted arguments for `comb`.

The function requires one or more swmpr objects as input as separate, undefined arguments. The remaining arguments must be called explicitly since an arbitrary number of objects can be used as input. In general, the function combines data by creating a master time series that is used to iteratively merge all swmpr objects. The time series for merging depends on the value passed to the `method` argument. Passing `'union'` to `method` will create a time series that is continuous starting from the earliest date and the latest date for all input objects. Passing `'intersect'` to `method` will create a time series that is continuous from the set of dates that are shared between all input objects. Finally, a seven or eight character station name passed to `method` will merge all input objects based

on a continuous time series for the given station. The specified station must be present in the input data. Currently, combining data types from different stations is not possible, excluding weather data which are typically at a single, dedicated station.

Value

Returns a combined swmpr object

See Also

setstep

Examples

```
## get nuts, wq, and met data as separate objects for the same station
## note that most sites usually have one weather station
path <- system.file('zip_ex', package = 'SWMPr')
swmp1 <- import_local(path, 'apacpnut')
swmp2 <- import_local(path, 'apacpwq')
swmp3 <- import_local(path, 'apaebmet')

## combine nuts and wq data by union
comb(swmp1, swmp2, method = 'union')

## combine nuts and wq data by intersect
comb(swmp1, swmp3, method = 'intersect')

## combine nuts, wq, and met data by nuts time series, two hour time step
comb(swmp1, swmp2, swmp3, timestep = 120, method = 'apacpnut')
```

---

decomp *Seasonal trend decomposition of swmpr data*

Description

Decompose swmpr data into trend, seasonal, and random components using decompose and ts

Usage

```
decomp(swmpr_in, ...)

## S3 method for class 'swmpr'
decomp(swmpr_in, param, type = "additive",
  frequency = "daily", start = NULL, ...)
```

Arguments

swmpr_in	input swmpr object
...	arguments passed to decompose, ts, and other methods
param	chr string of swmpr parameter to decompose
type	chr string of 'additive' or 'multiplicative' indicating the type of decomposition, default 'additive'.

frequency	chr string or numeric vector indicating the periodic component of the input parameter. Only 'daily' or 'seasonal' are accepted as chr strings. Otherwise a numeric vector specifies the number of observations required for a full cycle of the input parameter. Defaults to 'daily' for a diurnal parameter.
start	numeric vector indicating the starting value for the time series given the frequency. Only required if frequency is numeric. See ts.

Details

This function is a simple wrapper to the decompose function. The decompose function separates a time series into additive or multiplicative components describing a trend, cyclical variation (e.g., daily or seasonal), and the remainder. The additive decomposition assumes that the cyclical component of the time series is stationary (i.e., the variance is constant), whereas a multiplicative decomposition accounts for non-stationarity. By default, a moving average with a symmetric window is used to filter the seasonal component. Alternatively, a vector of filter coefficients in reverse time order can be supplied (see decompose).

The decompose function requires a ts object with a specified frequency. The decomp function converts the input swmpr vector to a ts object prior to decompose. This requires an explicit input defining the frequency in the time series required to complete a full period of the parameter. For example, the frequency of a parameter with diurnal periodicity would be 96 if the time step is 15 minutes (4 * 24). The frequency of a parameter with seasonal periodicity would be 35040 (4 * 24 * 365). For simplicity, chr strings of 'daily' or 'seasonal' can be supplied in place of numeric values. A starting value of the time series must be supplied in the latter case. Use of the setstep function is required to standardize the time step prior to decomposition.

Note that the decompose function is a relatively simple approach and alternative methods should be investigated if a more sophisticated decomposition is desired.

Value

Returns a decomposed.ts object

References

M. Kendall and A. Stuart (1983) The Advanced Theory of Statistics, Vol. 3, Griffin. pp. 410-414.

See Also

decompose, ts, stl

Examples

```
path <- system.file('zip_ex', package = 'SWMPr')

## get data, qaqc
swmp1 <- import_local(path, 'apadbwq')

## subset for daily decomposition
dat <- subset(swmp1, subset = c('2013-07-01 00:00', '2013-07-31 00:00'))

## decomposition and plot
test <- decomp(dat, param = 'do_mgl', frequency = 'daily')
plot(test)

## dealing with missing values
```

```
dat <- subset(swmp1, subset = c('2013-06-01 00:00', '2013-07-31 00:00'))

## this returns an error
## Not run:
test <- decomp(dat, param = 'do_mgl', frequency = 'daily')

## End(Not run)

## how many missing values?
sum(is.na(dat$do_mgl))

## use na.approx to interpolate missing data
dat <- na.approx(dat, params = 'do_mgl', maxgap = 10)

## decomposition and plot
test <- decomp(dat, param = 'do_mgl', frequency = 'daily')
plot(test)
```

`hist.swmpr` *Plot swmpr using a histogram*

Description

Plot a histogram showing the distribution of a swmpr parameter

Usage

```
## S3 method for class 'swmpr'
hist(x, subset = NULL, select, operator = NULL, ...)
```

Arguments

`x`	input swmpr object
`subset`	chr string of form 'YYYY-mm-dd HH:MM' to subset a date range. Input can be one (requires `operator` or two values (a range), passed to `subset`.
`select`	chr string of parameters to keep, passed to `subset`.
`operator`	chr string specifiying binary operator (e.g., `'>'`, `'<='`) if subset is one date value, passed to `subset`.
`...`	other arguments passed to `hist`

Details

The swmpr method for histograms is a convenience function for the default histogram function. Conventional histogram methods also work well since swmpr objects are also data frames. See the examples for use with different methods.

See Also

`hist`

Examples

```
## get data
path <- system.file('zip_ex', package = 'SWMPr')
dat <- import_local(path, 'apadbwq')

## plot using swmpr method, note default line plot
hist(dat, select = 'do_mgl')

## plot using defualt method
hist(dat$do_mgl)
```

`import_local` *Import local CDMO data*

Description

Import local data that were obtained from the CDMO through the zip downloads feature

Usage

```
import_local(path, station_code, trace = FALSE)
```

Arguments

`path`	chr string of full path to .csv files with raw data
`station_code`	chr string of station to import, typically 7 or 8 characters but may include full name with year, excluding file extension
`trace`	logical indicating if progress is sent to console, default `FALSE`

Details

The function is designed to import local data that were downloaded from the CDMO outside of R. This approach works best for larger data requests. The function is designed for data from the zip downloads feature in the advanced query section of the CDMO. The function may also work using data from the data export system, but this feature has not been extensively tested (expect bugs). The downloaded data will be in a compressed folder that includes multiple .csv files by year for a given data type (e.g., apacpwq2002.csv, apacpwq2003.csv, apacpnut2002.csv, etc.). The import_local function can be used after the folder is decompressed.

Zip download request through CDMO: http://cdmo.baruch.sc.edu/aqs/zips.cfm

Value

Returns a swmpr object with all parameters and QAQC columns for the station. The full date range in the raw data are also imported.

See Also

`all_params`, `all_params_dtrng`, `single_param`

Examples

```
## this is the path for csv example files
path <- system.file('zip_ex', package = 'SWMPr')

## import, do not include file extension
import_local(path, 'apaebmet')
```

`map_reserve` *Map a reserve*

Description

Create a map of all the stations in a reserve

Usage

```
map_reserve(nerr_site_id, zoom = 11, text_sz = 6, text_col = "black",
  map_type = "terrain")
```

Arguments

`nerr_site_id`	chr string of the reserve to map, first three characters used by NERRS
`zoom`	numeric value for map zoom, passed to `get_map`
`text_sz`	numeric value for text size of station names, passed to `geom_text`
`text_col`	chr string for text color of station names, passed to `geom_text`
`map_type`	chr string indicating the type of base map obtained from Google maps, values are `terrain` (default), `satellite`, `roadmap`, or `hybrid`

Details

This function is a simple wrapper to functions in the ggmap package which returns a map of all of the stations at a NERRS reserve. The `zoom` argument may have to be chosen through trial and error depending on the spatial extent of the reserve. Additionally, station locations are returned using the `site_codes_ind` function if the computer making the request has the IP address registered with CDMO. Otherwise, a local and possibly outdated file is used. See the package documentation.

Value

A `ggplot` object for plotting.

See Also

`get_map`, `ggmap`, `ggplot`, `site_codes_ind`

Examples

```
## defaults

map_reserve('jac')

## satellite map

map_reserve('jac', map_type = 'satellite')

## roadmap

map_reserve('jac', map_type = 'roadmap')

## hybrid

map_reserve('jac', map_type = 'hybrid')
```

`na.approx.swmpr` *Linearly interpolate gaps*

Description

Linearly interpolate gaps in swmpr data within a maximum size

Usage

```
## S3 method for class 'swmpr'
na.approx(object, params = NULL, maxgap, ...)
```

Arguments

`object`	input swmpr object
`params`	is chr string of swmpr parameters to smooth, default all
`maxgap`	numeric vector indicating maximum gap size to interpolate where size is numer of records, must be explicit
`...`	additional arguments passed to other methods

Details

A common approach for handling missing data in time series analysis is linear interpolation. A simple curve fitting method is used to create a continuous set of records between observations separated by missing data. A required argument for the function is `maxgap` which defines the maximum gap size for interpolation. The ability of the interpolated data to approximate actual, unobserved trends is a function of the gap size. Interpolation between larger gaps are less likely to resemble patterns of an actual parameter, whereas interpolation between smaller gaps may be more likely to resemble actual patterns. An appropriate gap size limit depends on the unique characteristics of specific datasets or parameters.

Value

Returns a swmpr object. QAQC columns are removed if included with input object.

See Also

na.approx

Examples

```
## import data
path <- system.file('zip_ex', package = 'SWMPr')
swmp1 <- import_local(path, 'apadbwq')

## qaqc and subset imported data
dat <- qaqc(swmp1)
dat <- subset(dat, subset = c('2013-01-22 00:00', '2013-01-26 00:00'))

## interpolate, maxgap of 10 records
test <- na.approx(dat, params = 'do_mgl', maxgap = 10)

## interpolate maxgap of 30 records
test2 <- na.approx(dat, params = 'do_mgl', maxgap = 30)

## plot for comparison
par(mfrow = c(3,1))

plot(do_mgl ~ datetimestamp, dat, main = 'Raw', type = 'l')

plot(do_mgl ~ datetimestamp, test, col = 'red',
 main = 'Inteprolation - maximum gap of 10 records',
 type = 'l')
lines(dat, select = 'do_mgl')

plot(do_mgl ~ datetimestamp, test2, col = 'red',
 main = 'Interpolation - maximum gap of 30 records',
 type = 'l')
lines(dat, select = 'do_mgl')
```

---

param_names *Get parameters of a given type*

Description

Get parameter column names for each parameter type

Usage

```
param_names(param_type = c("nut", "wq", "met"))
```

Arguments

param_type	chr string specifying 'nut', 'wq', or 'met'. Input can be one to three types.

Details

This function is used internally within several functions to return a list of the expected parameters for a given parameter type: nutrients, water quality, or meteorological. It does not need to be called explicitly.

Value

Returns a named list of parameters for the `param_type`. The parameter names are lower-case strings of SWMP parameters and corresponding qaqc names (`'f_'` prefix)

`parser` *Parse SOAP objects for swmpr*

Description

Parsing function for objects returned from SOAP server

Usage

```
parser(soap_in, parent_in = "data")
```

Arguments

`soap_in`	soap object returned from CDMO server
`parent_in`	chr string of parent nodes to parse

Details

This function parses XML objects returned from the CDMO SOAP server, which are further passed to `swmpr`. It is used internally by the data retrieval functions, excluding `import_local`. The function does not need to be called explicitly.

Value

Returns a `data.frame` of parsed XML nodes

`plot.swmpr` *Plot swmpr data*

Description

Plot a time series of parameters in a swmpr object

Usage

```
## S3 method for class 'swmpr'
plot(x, type = "l", ...)

## S3 method for class 'swmpr'
lines(x, subset = NULL, select, operator = NULL, ...)
```

Arguments

x	input swmpr object
type	chr string for type of plot, default 'l'. See plot.
...	other arguments passed to par, plot.default, lines, points
subset	chr string of form 'YYYY-mm-dd HH:MM' to subset a date range. Input can be one (requires operator or two values (a range). Passed to subset.swmpr.
select	chr string of parameters to keep. Passed to subset.swmpr.
operator	chr string specifiying binary operator (e.g., '>', '<=') if subset is one date value. Passed to subset.swmpr.

Details

The swmpr method for plotting is a convenience function for plotting a univariate time series. Conventional plotting methods also work well since swmpr objects are also data frames. See the examples for use with different methods.

See Also

plot

Examples

```
## get data
path <- system.file('zip_ex', package = 'SWMPr')
swmp1 <- import_local(path, 'apadbwq')

## subset
dat <- subset(swmp1, select = 'do_mgl', subset = c('2013-07-01 00:00', '2013-07-31 00:00'))

## plot using swmpr method, note default line plot
plot(dat)

## plot using formula method
plot(do_mgl ~ datetimestamp, dat)

## plot using defualt, add lines
plot(dat$datetimestamp, dat$do_mgl)
lines(dat, select = 'do_mgl', col = 'red')
```

---

qaqc *QAQC filtering for SWMP data*

Description

QAQC filtering for SWMP data obtained from retrieval functions, local and remote

Usage

```
qaqc(swmpr_in, ...)

## S3 method for class 'swmpr'
qaqc(swmpr_in, qaqc_keep = 0, trace = FALSE, ...)
```

Arguments

swmpr_in	input swmpr object
...	arguments passed to or from other methods
qaqc_keep	numeric vector of qaqc flags to keep, default 0
trace	logical for progress output on console, default FALSE

Details

The qaqc function is a simple screen to retain values from the data with specified QAQC flags, described online: http://cdmo.baruch.sc.edu/data/qaqc.cfm. Each parameter in the swmpr data typically has a corresponding QAQC column of the same name with the added prefix 'f_'. Values in the QAQC column specify a flag from -5 to 5. Generally, only data with the '0' QAQC flag should be used, which is the default option for the function. Data that do not satisfy QAQC criteria are converted to NA values. Processed data will have QAQC columns removed, in addition to removal of values in the actual parameter columns that do not meet the criteria.

Value

Returns a swmpr object with NA values for records that did not match qaqc_keep. QAQC columns are also removed.

See Also

qaqcchk

Examples

```
## get data
path <- system.file('zip_ex', package = 'SWMPr')
dat <- import_local(path, 'apadbwq')

## qaqc screen for a swmpr object, retain only '0'
qaqc(dat)

## retain all data regardless of flag
qaqc(dat, qaqc_keep = NULL)

## retain only '0' and '-1' flags
qaqc(dat, qaqc_keep = c(0, -1))
```

---

qaqcchk *Summary of QAQC flags in SWMP data*

Description

Summary of the number of observations with a given QAQC flag for each parameter column of a swmpr object

Usage

```
qaqcchk(swmpr_in)

## S3 method for class 'swmpr'
qaqcchk(swmpr_in)
```

Arguments

swmpr_in	input swmpr object

Details

Viewing the number of observations for each parameter that are assigned to a QAQC flag may be useful for deciding how to process the data qith qaqc. The qaqcchk function can be used to view this information. Consult the online documentation for a description of each QAQC flag: http://cdmo.baruch.sc.edu/data/qaqc.cfm

Value

Returns a data.frame with columns for swmpr parameters and row counts indicating the number of observations in each parameter assigned to a flag value.

See Also

qaqc

Examples

```
## get data
path <- system.file('zip_ex', package = 'SWMPr')
dat <- import_local(path, 'apadbwq')

## view the number observations in each QAQC flag
qaqcchk(dat)
```

---

setstep *Format a swmpr time vctor*

Description

Create a continuous time vector at set time step for a swmpr object

Usage

```
setstep(swmpr_in, ...)

## S3 method for class 'swmpr'
setstep(swmpr_in, timestep = 15, differ = timestep/2, ...)
```

Arguments

swmpr_in	input swmpr object
...	arguments passed to or from other methods
timestep	numeric value of time step to use in minutes
differ	numeric value defining buffer for merging time stamps to standardized time series

Details

The setstep function formats a swmpr object to a continuous time series at a given time step. This function is not necessary for most stations but can be useful for combining data or converting an existing time series to a set interval. The first argument, timestep, specifies the desired time step in minutes starting from the nearest hour of the first observation. The second argument, differ, specifies the allowable tolerance in minutes for matching existing observations to user-defined time steps in cases where the two are dissimilar. Values for differ that are greater than one half the value of timestep are not allowed to prevent duplication of existing data. Likewise, the default value for differ is one half the time step. Rows that do not match any existing data within the limits of the differ argument are not discarded. Output from the function can be used with subset and to create a time series at a set interval with empty data removed.

Value

Returns a swmpr object for the specified time step

See Also

comb

Examples

```
## import data
path <- system.file('zip_ex', package = 'SWMPr')
dat <- import_local(path, 'apaebmet')

## convert time series to two hour invervals
## tolerance of +/- 30 minutes for matching existing data
setstep(dat, timestep = 120, differ = 30)

## convert a nutrient time series to a continuous time series
## then remove empty rows and columns
dat_nut <- import_local(path, 'apacpnut')
dat_nut <- setstep(dat_nut, timestep = 60)
subset(dat_nut, rem_rows = TRUE, rem_cols = TRUE)
```

---

single_param *Get CDMO records for a single parameter*

Description

Get stations records from the CDMO for a single parameter starting with the most current date

Usage

```
single_param(station_code, param, Max = 100)
```

Arguments

station_code	chr string of station, 7 or 8 characters
param	chr string for a single parameter to return.
Max	numeric value for number of records to obtain from the current date, maximum of 100

Details

This function uses the SSOAP package to retrieve data from the CDMO through a SOAP client interface. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: http://cdmo.baruch.sc.edu/webservices.cfm. This function is the CDMO equivalent of exportSingleParamXML.

Value

Returns a swmpr object with one parameter. QAQC columns are not provided.

See Also

all_params, all_params_dtrng

Examples

```
## Not run:

## single parameter for a station, most recent
single_param('hudscwq', 'do_mgl')


## End(Not run)
```

---

site_codes *Obtain metadata for all stations*

Description

Obtain a data.frame of metadata for all SWMP stations.

Usage

```
site_codes()
```

Details

This function uses the SSOAP package to retrieve data from the CDMO through a SOAP client interface. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: http://cdmo.baruch.sc.edu/webservices.cfm. This is the CDMO equivalent of exportStationCodesXML.

Value

A data.frame of SWMP metadata

Examples

```
## Not run:

## retrieve metadata for all sites
site_codes()


## End(Not run)
```

---

site_codes_ind *Obtain metadata for a single reserve*

Description

Get metadata for all the stations at a single SWMP reserve

Usage

```
site_codes_ind(nerr_site_id)
```

Arguments

nerr_site_id	chr string of site, three letters

Details

This function uses the SSOAP package to retrieve data from the CDMO through a SOAP client interface. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: http://cdmo.baruch.sc.edu/webservices.cfm. This function is the CDMO equivalent of NERRFilterStationCodesXMLNew.

Value

An abbreviated data.frame of the SWMP metadata for the requested site

Examples

```
## Not run:

## retrieve metadata for all stations at a site
site_codes_ind('apa')


## End(Not run)
```

smoother *Smooth swmpr data*

Description

Smooth swmpr data with a moving window average

Usage

```
smoother(swmpr_in, ...)

## S3 method for class 'swmpr'
smoother(swmpr_in, window = 5, sides = 2, params = NULL,
  ...)
```

Arguments

swmpr_in	input swmpr object
...	arguments passed to or from other methods
window	numeric vector of ones defining size of smoothing window, passed to filter
sides	numeric vector defining method of averaging, passed to filter
params	is chr string of swmpr parameters to smooth, default all

Details

The smoother function can be used to smooth parameters in a swmpr object using a specified window size. This method is a simple wrapper to filter. The window argument specifies the number of observations included in the moving average. The sides argument specifies how the average is calculated for each observation (see the documentation for filter). A value of 1 will filter observations within the window that are previous to the current observation, whereas a value of 2 will filter all observations within the window centered at zero lag from the current observation. The params argument specifies which parameters to smooth.

Value

Returns a filtered swmpr object. QAQC columns are removed if included with input object.

See Also

filter

Examples

```
## import data
path <- system.file('zip_ex', package = 'SWMPr')
swmp1 <- import_local(path, 'apadbwq')

## qaqc and subset imported data
dat <- qaqc(swmp1)
dat <- subset(dat, subset = c('2012-07-09 00:00', '2012-07-24 00:00'))
```

```
## filter
test <- smoother(dat, window = 50, params = 'do_mgl')

## plot to see the difference
plot(do_mgl ~ datetimestamp, data = dat, type = 'l')
lines(test, select = 'do_mgl', col = 'red', lwd = 2)
```

`subset.swmpr` *Subset a swmpr object*

Description

Subset a swmpr object by a date range, parameters, or non-empty values

Usage

```
## S3 method for class 'swmpr'
subset(x, subset = NULL, select = NULL, operator = NULL,
  rem_rows = FALSE, rem_cols = FALSE, ...)
```

Arguments

`x`	input swmpr object
`subset`	chr string of form 'YYYY-mm-dd HH:MM' to subset a date range. Input can be one (requires `operator` or two values (a range).
`select`	chr string of parameters to keep
`operator`	chr string specifiying binary operator (e.g., `'>'`, `'<='`) if subset is one date value
`rem_rows`	logical indicating if rows with no data are removed, default `FALSE`
`rem_cols`	is logical indicating if cols with no data are removed, default `FALSE`
`...`	arguments passed to other methods

Details

This function is used to subset swmpr data by date and/or a selected parameter. The date can be a single value or as two dates to select records within the range. The former case requires a binary operator input as a character string passed to the argument, such as > or <. The subset argument for the date(s) must also be a character string of the format YYYY-mm-dd HH:MM for each element (i.e., ‘

Value

Returns a swmpr object as a subset of the input. The original object will be returned if no arguments are specified.

See Also

`subset`

Examples

```
## get data
path <- system.file('zip_ex', package = 'SWMPr')
dat <- import_local(path, 'apaebmet')

## select two parameters from dat
subset(dat, select = c('rh', 'bp'))

## subset records greater than or equal to a date
subset(dat, subset = '2013-01-01 0:00', operator = '>=')

## subset records within a date range
subset(dat, subset = c('2012-07-01 6:00', '2012-08-01 18:15'))

## subset records within a date range, select two parameters
subset(dat, subset = c('2012-07-01 6:00', '2012-08-01 18:15'),
   select = c('atemp', 'totsorad'))

## remove rows/columns that do not contain data
subset(dat, rem_rows = TRUE, rem_cols = TRUE)
```

---

swmpr *Create a wmpr object*

Description

Wrapper for creating a swmpr object

Usage

```
swmpr(stat_in, meta_in)
```

Arguments

stat_in	data.frame of swmp data
meta_in	chr string for station code (7 or 8 characters), can be multiple stations if data are combined

Details

This function is a simple wrapper to `structure` that is used internally within other functions to create a swmpr object. The function does not have to be used explicitly.

Value

Returns a swmpr object to be used with S3 methods

time_vec *Format SWMP datetimestamp*

Description

Format the datetimestamp column of SWMP data

Usage

```
time_vec(chr_in = NULL, station_code, tz_only = FALSE)
```

Arguments

chr_in	chr string of datetimestamp vector
station_code	is chr string for station (three or more characters)
tz_only	logical that returns only the timezone, default FALSE

Details

This function formats the datetimestamp column of SWMP data to the POSIXct format and the correct timezone for a station. Note that SWMP data do not include daylight savings and the appropriate location based on GMT offsets is used for formatting. This function is used internally within data retrieval functions and does not need to be called explicitly.

Value

Returns a POSIX vector if tz_only is true, otherwise the timezone for a station is returned as a chr string

Index