Package: SWMPr (via r-universe)

October 28, 2026

Type Package

Title Retrieving, Organizing, and Analyzing Estuary Monitoring Data

Version 2.5.3

Date 2026-10-28

Author Marcus W. Beck [aut, cre], Kimberly Cressman [ctb]

Maintainer Marcus W. Beck <mbeck@tbep.org>

Description Tools for retrieving, organizing, and analyzing environmental data from the System Wide Monitoring Program of the National Estuarine Research Reserve System <https://cdmo.baruch.sc.edu/>. These tools address common challenges associated with continuous time series data for environmental decision making.

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

License CC0

Imports data.table, httr, gridExtra, oce, dplyr, lattice, openair, RColorBrewer, reshape2, tictoc, tidyr, XML

LazyData true

Depends R (>= 3.2.0), ggplot2, zoo

Suggests colorspace, covr, ggmap, knitr, rmarkdown, testthat (>= 3.0.0)

RoxygenNote 7.3.3

VignetteBuilder knitr

URL http://fawda123.github.io/SWMPr/

Config/testthat/edition 3

NeedsCompilation no

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

Date/Publication 2026-10-28 13:30:02 UTC

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

RemoteRef HEAD

RemoteSha 4e7cd9d73332e242d73f94c840951a1ec48d1761

Contents

aggremetab . . . 3
aggreswmp . . . 4
all_params . . . 6
all_params_dtrng . . . 7
apacpnut . . . 8
apacpwq . . . 9
apadbwq . . . 10
apaebmet . . . 11
calckl . . . 13
cens_id . . . 14
comb . . . 15
decomp . . . 16
decomp_cj . . . 19
decompTs . . . 21
ecometab . . . 22
gradcols . . . 25
hist.swmpr . . . 26
import_local . . . 27
map_reserve . . . 29
metab_day . . . 30
na.approx.swmpr . . . 31
overplot . . . 32
oxySol . . . 34
param_names . . . 35
parser . . . 36
plot.swmpr . . . 36
plot_metab . . . 37
plot_quants . . . 39
plot_summary . . . 40
plot_wind . . . 42
qaqc . . . 44
qaqcchk . . . 45
rem_reps . . . 46
setstep . . . 47
single_param . . . 49
site_codes . . . 50
site_codes_ind . . . 50
smoother . . . 51
stat_locs . . . 52
subset.swmpr . . . 53
sun_times . . . 54
swmpr . . . 55
time_vec . . . 56

Index **57**

aggremetab *Aggregate metabolism data*

Description

Aggregate a metabolism attribute from swmpr data by a specified time period and method

Usage

```
aggremetab(swmpr_in, ...)

## S3 method for class 'swmpr'
aggremetab(swmpr_in, by = "weeks", na.action = na.pass, alpha = 0.05, ...)
```

Arguments

swmpr_in	input swmpr object
...	additional arguments passed to other methods
by	chr string or numeric value specifying aggregation period. If chr string, must be 'years', 'quarters', 'months', 'weeks', 'days', or 'hours'. A numeric value indicates the number of days for a moving window average. Additional arguments passed to smoother can be used if by is numeric.
na.action	function for treating missing data, default na.pass
alpha	numeric indicating alpha level of confidence interval for aggregated data

Details

The function summarizes metabolism data by averaging across set periods of observation. Confidence intervals are also returned based on the specified alpha level. It is used within plot_metab function to view summarized metabolism results. Data can be aggregated by 'years', 'quarters', 'months', or 'weeks' for the supplied function, which defaults to the mean. 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.

Value

Returns an aggregated metabolism data.frame if the metabolism attribute of the swmpr object is not NULL. Upper and lower confidence limits are also provided if the aggregation period was specified as a character string.

See Also

aggregate, aggreswmp, ecometab, plot_metab

Examples

```
## Not run:
## import water quality and weather data
data(apadbwq)
data(apaebmet)

## qaqc, combine
wq <- qaqc(apadbwq)
met <- qaqc(apaebmet)
dat <- comb(wq, met)

## estimate metabolism
res <- ecometab(dat)

## change aggregation period and alpha
aggremetab(res, by = 'months', alpha = 0.1)

## use a moving window average of 30 days
aggremetab(res, by = 30)

## use a left-centered window instead
aggremetab(res, by = 30, sides = 1)

## End(Not run)
```

---

aggreswmp *Aggregate swmpr data*

Description

Aggregate swmpr data by specified time period and method

Usage

```
aggreswmp(swmpr_in, ...)

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

Arguments

swmpr_in	input swmpr object
...	additional arguments passed to other methods
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
plot	logical to return a plot of the summarized data, default FALSE
na.action	function for treating missing data, default na.pass. See the documentation for aggregate for options.

Details

The function aggregates 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. If aggs_out = TRUE, the original swmpr object is returned with the datetimestamp column formatted for the first day of the aggregation period from by. A ggplot object of boxplot summaries is returned if plot = TRUE.

See Also

aggregate

Examples

```
## Not run:
## get data, prep
data(apacpwq)
dat <- apacpwq
swmpr_in <- subset(qaqc(dat), rem_cols = TRUE)

## get mean DO by quarters
```

```
aggreswmp(swmpr_in, 'quarters', params = c('do_mgl'))

## get a plot instead
aggreswmp(swmpr_in, 'quarters', params = c('do_mgl'), plot = T)

## plots with other variables
p <- aggreswmp(swmpr_in, 'months', params = c('do_mgl', 'temp', 'sal'), plot = T)
p
library(ggplot2)
p + geom_boxplot(aes(fill = var)) + theme(legend.position = 'none')

## 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)
aggreswmp(swmpr_in, FUN = fun_in, 'years')

## End(Not run)
```

`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, param = NULL, trace = TRUE)
```

Arguments

`station_code`	chr string of station, 7 or 8 characters
`Max`	numeric value for number of records to obtain from the current date
`param`	chr string for a single parameter to return, defaults to all parameters for a station type.
`trace`	logical indicating if import progress is printed in console

Details

This function retrieves data from the CDMO through the web services URL. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: https://cdmo.baruch.sc.edu/webservices.cfm. Function is the CDMO equivalent of `exportAllParamsXMLNew` but actually uses `all_params_dtrng`, which is a direct call to `exportAllParamsDateRangeXMLNew`.

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, trace = TRUE, Max = 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.
`trace`	logical indicating if import progress is printed in console
`Max`	numeric indicating maximum number of records to return

Details

This function retrieves data from the CDMO through the web services URL. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: https://cdmo.baruch.sc.edu/webservices.cfm. This function is the CDMO equivalent of `exportAllParamsDateRangeXMLNew`. Download time may be excessive for large requests.

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.

See Also

all_params, single_param

Examples

```
## Not run:

## get all parameters within a date range
all_params_dtrng('apaebwq', c('01/01/2013', '02/01/2013'))

## get single parameter within a date range
all_params_dtrng('apaebwq', c('01/01/2013', '02/01/2013'),
   param = 'do_mgl')


## End(Not run)
```

---

apacpnut *Example nutrient data for Apalachicola Bay Cat Point station.*

Description

An example nutrient dataset for Apalachicola Bay Cat Point station. The data are a swmpr object that have been imported into R from csv files using the import_local function. The raw data were obtained from the CDMO data portal but can also be accessed from a zip file created for this package. See the source below. The help file for the import_local function describes how the data can be imported from the zip file. Attributes of the dataset include names, row.names, class, station, parameters, qaqc_cols, date_rng, timezone, and stamp_class.

Usage

```
apacpnut
```

Format

A swmpr object and data.frame with 215 rows and 13 variables:

datetimestamp POSIXct

po4f num

f_po4f chr

nh4f num

f_nh4f chr

no2f num

f_no2f chr

no3f num

f_no3f chr

no23f num

f_no23f chr

chla_n num

f_chla_n chr

Source

https://s3.amazonaws.com/swmpexdata/zip_ex.zip

Examples

```
data(apacpnut)
```

---

apacpwq *Example water quality data for Apalachicola Bay Cat Point station.*

Description

An example water quality dataset for Apalachicola Bay Cat Point station. The data are a swmpr object that have been imported into R from csv files using the import_local function. The raw data were obtained from the CDMO data portal but can also be accessed from a zip file created for this package. See the source below. The help file for the import_local function describes how the data can be imported from the zip file. Attributes of the dataset include names, row.names, class, station, parameters, qaqc_cols, date_rng, timezone, and stamp_class.

Usage

```
apacpwq
```

Format

A swmpr object and data.frame with 70176 rows and 25 variables:

datetimestamp POSIXct

temp num

f_temp chr

spcond num

f_spcond chr

sal num

f_sal chr

do_pct num

f_do_pct chr

do_mgl num

f_do_mgl chr

depth num

f_depth chr

cdepth num

f_cdepth chr

level num

f_level chr

clevel num

f_clevel chr

ph num

f_ph chr

turb num

f_turb chr

chlfluor num

f_chlfluor chr

Source

https://s3.amazonaws.com/swmpexdata/zip_ex.zip

Examples

```
data(apacpwq)
```

---

apadbwq *Example water quality data for Apalachicola Bay Dry Bar station.*

Description

An example water quality dataset for Apalachicola Bay Dry Bar station. The data are a swmpr object that have been imported into R from csv files using the import_local function. The raw data were obtained from the CDMO data portal but can also be accessed from a zip file created for this package. See the source below. The help file for the import_local function describes how the data can be imported from the zip file. Attributes of the dataset include names, row.names, class, station, parameters, qaqc_cols, date_rng, timezone, and stamp_class.

Usage

```
apadbwq
```

Format

A swmpr object and data.frame with 70176 rows and 25 variables:

datetimestamp POSIXct

temp num

f_temp chr

spcond num

f_spcond chr

sal num

f_sal chr
do_pct num
f_do_pct chr
do_mgl num
f_do_mgl chr
depth num
f_depth chr
cdepth num
f_cdepth chr
level num
f_level chr
clevel num
f_clevel chr
ph num
f_ph chr
turb num
f_turb chr
chlfluor num
f_chlfluor chr

Source

https://s3.amazonaws.com/swmpexdata/zip_ex.zip

Examples

```
data(apadbwq)
```

---

apaebmet *Example weather data for Apalachicola Bay East Bay station.*

Description

An example weather dataset for Apalachicola Bay East Bay station. The data are a swmpr object that have been imported into R from csv files using the import_local function. The raw data were obtained from the CDMO data portal but can also be accessed from a zip file created for this package. See the source below. The help file for the import_local function describes how the data can be imported from the zip file. Attributes of the dataset include names, row.names, class, station, parameters, qaqc_cols, date_rng, timezone, and stamp_class.

Usage

```
apaebmet
```

Format

A swmpr object and data.frame with 70176 rows and 23 variables:

datetimestamp POSIXct

atemp num

f_atemp chr

rh num

f_rh chr

bp num

f_bp chr

wspd num

f_wspd chr

maxwspd num

f_maxwspd chr

wdir num

f_wdir chr

sdwdir num

f_sdwdir chr

totpar num

f_totpar chr

totprcp num

f_totprcp chr

cumprcp num

f_cumprcp chr

totsorad num

f_totsorad chr

Source

https://s3.amazonaws.com/swmpexdata/zip_ex.zip

Examples

```
data(apaebmet)
```

calckl *Calculate oxygen mass transfer coefficient*

Description

Calculate oxygen mass transfer coefficient using equations in Thiebault et al. 2008. Output is used to estimate the volumetric reaeration coefficient for ecosystem metabolism.

Usage

```
calckl(temp, sal, atemp, wspd, bp, height = 10)
```

Arguments

temp	numeric for water temperature (C)
sal	numeric for salinity (ppt)
atemp	numeric for air temperature (C)
wspd	numeric for wind speed (m/s)
bp	numeric for barometric pressure (mb)
height	numeric for height of anemometer (meters)

Details

This function is used within the ecometab function and should not be used explicitly.

Value

Returns numeric value for oxygen mass transfer coefficient (m d-1).

References

Ro KS, Hunt PG. 2006. A new unified equation for wind-driven surficial oxygen transfer into stationary water bodies. Transactions of the American Society of Agricultural and Biological Engineers. 49(5):1615-1622.

Thebault J, Schraga TS, Cloern JE, Dunlavey EG. 2008. Primary production and carrying capacity of former salt ponds after reconnection to San Francisco Bay. Wetlands. 28(3):841-851.

See Also

ecometab

cens_id *Flag observations above/below detection limits*

Description

Flag observations above/below detection limits

Usage

```
cens_id(swmpr_in, ...)

## S3 method for class 'swmpr'
cens_id(swmpr_in, flag_type = "both", select = NULL, ...)
```

Arguments

swmpr_in	input swmpr object
...	optional arguments passed to or from other methods
flag_type	chr string indicating the flag type to return, must be one of 'below', 'above', or 'both', see details
select	chr string of parameters to keep, defaults to all, 'datetimestamp' will always be kept

Details

Censored observations are identified in swmpr objects using the CDMO flags -4 or -5, indicating outside the low or high sensor range, respectively. Additional codes are identified including A (-2007) or SUL (2007-) for above and B (-2007), SBL (2007-), SCB (2007-, calculated) for below detection limits. The QAQC columns are searched for all parameters and replaced with the appropriate value indicating the detection limit as defined by flag_type. The default argument flag_type = 'both' will recode the QAQC columns as -1, 0, or 1 indicating below, within, or above the detection limit. Setting flag_type = 'below' or 'above' will convert the columns to TRUE/FALSE values indicating observations beyond the detection limit (either above or below, TRUE) or within the normal detection range FALSE. The output includes additional columns similar to those for QAQC flags, such that the column names for censored flags include a c_ prefix before the parameter name. Note that the function will of course not work if already processed with qaqc. QAQC columns are retained for additional processing.

The user should refer to the metadata or visually examine the observed data to identify the actual limit, which may change over time.

Value

Returns a swmpr object with additional columns for censored flag values and the appropriate flag type based on the input arguments. Censored flag columns are named with a c_ prefix.

See Also

qaqc

Examples

```
## get data
data(apacpnut)
dat <- apacpnut

## convert all qaqc columns to censored flags, -1 below, 0 within, 1 above
cens_id(dat)

## T/F for above or within, note that none are above
cens_id(dat, flag_type = 'above')

## T/F for below or within
cens_id(dat, flag_type = 'below')
```

---

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 = NULL, method = "union")

## Default S3 method:
comb(..., date_col, timestep = 15, differ = NULL, method = "union")
```

Arguments

`...`	input time series data objects, 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 data. Use `'union'` for all dates as continuous time series or `'intersect'` for only areas of overlap. If input is a `swmpr` object, a `'station'` name can be used to combine by the date range of a given station, assuming there is overlap with the second station. A numeric value can be supplied for the default method that specifies which data object to use for the date range based on order of execution in the function call.
`date_col`	chr string indicating name of the date column

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 wq and met data as separate objects for the same station
swmp1 <- apacpnut
swmp2 <- apaebmet

## combine nuts and wq data by union, set timestep to 120 minutes
## Not run:
comb(swmp1, swmp2, timestep = 120, method = 'union')

## End(Not run)
```

---

decomp *Simple trend decomposition*

Description

Decompose data into trend, cyclical (e.g., daily, annual), and random components using decompose and ts

Usage

```
decomp(dat_in, ...)

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

## Default S3 method:
decomp(
  dat_in,
  param,
  date_col,
  type = "additive",
  frequency = "daily",
  start = NULL,
  ...
)
```

Arguments

dat_in	input data 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 'annual' 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.
date_col	chr string of the name of the date column

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 annual), 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 cyclical 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 (24 hours * 60 minutes / 15 minutes). The frequency of a parameter with annual periodicity at a 15 minute time step would be 35040 (365 days * 24 hours * 60 minutes / 15 minutes). For simplicity, chr strings of 'daily' or 'annual' 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

```
## get data
data(apadbwq)
swmp1 <- 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)
```

---

decomp_cj *Simple trend decomposition of monthly swmpr data*

Description

Decompose monthly SWMP time series into grandmean, annual, seasonal, and event series as described in Cloern and Jassby 2010.

Usage

```
decomp_cj(dat_in, ...)

## S3 method for class 'swmpr'
decomp_cj(
  dat_in,
  param,
  vals_out = FALSE,
  event = TRUE,
  type = c("add", "mult"),
  center = c("mean", "median"),
  ...
)

## Default S3 method:
decomp_cj(
  dat_in,
  param,
  date_col,
  vals_out = FALSE,
  event = TRUE,
  type = c("add", "mult"),
  center = c("mean", "median"),
  ...
)
```

Arguments

dat_in	input data object
...	additional arguments passed to or from other methods
param	chr string of variable to decompose

vals_out	logical indicating of numeric output is returned, default is FALSE to return a plot.
event	logical indicating if an 'events' component should be determined
type	chr string indicating the type of decomposition, either additive ('add') or multiplicative ('mult')
center	chr string indicating the method of centering, either 'mean' or 'median'
date_col	chr string indicating the name of the date column which should be a date or POSIX object.

Details

This function is a simple wrapper to the decompTs function in the archived wq package, also described in Cloern and Jassby (2010). The function is similar to decomp.swmpr (which is a wrapper to decompose) with a few key differences. The decomp.swmpr function decomposes the time series into a trend, seasonal, and random components, whereas the current function decomposes into the grandmean, annual, seasonal, and events components. For both functions, the random or events components, respectively, can be considered anomalies that don't follow the trends in the remaining categories.

The decomp_cj function provides only a monthly decomposition, which is appropriate for characterizing relatively long-term trends. This approach is meant for nutrient data that are obtained on a monthly cycle. The function will also work with continuous water quality or weather data but note that the data are first aggregated on the monthly scale before decomposition. Use the decomp.swmpr function to decompose daily variation.

Value

A ggplot object if vals_out = FALSE (default), otherwise a monthly time series matrix of class ts.

References

Cloern, J.E., Jassby, A.D. 2010. Patterns and scales of phytoplankton variability in estuarine-coastal ecosystems. Estuaries and Coasts. 33:230-241.

See Also

ts

Examples

```
## get data
data(apacpnut)
dat <- apacpnut
dat <- qaqc(dat, qaqc_keep = NULL)

## decomposition of chl, values as data.frame
decomp_cj(dat, param = 'chla_n', vals_out = TRUE)

## decomposition of chl, ggplot
decomp_cj(dat, param = 'chla_n')
```

```
## decomposition changing arguments passed to decompTs
decomp_cj(dat, param = 'chla_n', type = 'mult')

## Not run:
## monthly decomposition of continuous data
data(apacpwq)
dat2 <- qaqc(apacpwq)

decomp_cj(dat2, param = 'do_mgl')

## using the default method with a data frame
dat <- data.frame(dat)
decomp_cj(dat, param = 'chla_n', date_col = 'datetimestamp')

## End(Not run)
```

`decompTs` *Decompose a time series*

Description

The function decomposes a time series into a long-term mean, annual, seasonal and "events" component. The decomposition can be multiplicative or additive, and based on median or mean centering. Function and documentation herein are from archived wq package.

Usage

```
decompTs(
  x,
  event = TRUE,
  type = c("add", "mult"),
  center = c("mean", "median")
)
```

Arguments

`x`	a monthly time series vector
`event`	whether or not an "events" component should be determined
`type`	the type of decomposition, either multiplicative ("mult") or additive ("add")
`center`	the method of centering, either median or mean

Details

The rationale for this simple approach to decomposing a time series, with examples of its application, is given by Cloern and Jassby (2010). It is motivated by the observation that many important events for estuaries (e.g., persistent dry periods, species invasions) start or stop suddenly. Smoothing to extract the annualized term, which can disguise the timing of these events and make analysis of them unnecessarily difficult, is not used.

A multiplicative decomposition will typically be useful for a biological community- or population-related variable (e.g., chlorophyll-a) that experiences exponential changes in time and is approximately lognormal, whereas an additive decomposition is more suitable for a normal variable. The default centering method is the median, especially appropriate for series that have large, infrequent events.

If `event = TRUE`, the seasonal component represents a recurring monthly pattern and the events component a residual series. Otherwise, the seasonal component becomes the residual series. The latter is appropriate when seasonal patterns change systematically over time.

Value

A monthly time series matrix with the following individual time series:

`original`	original time series
`annual`	annual mean series
`seasonal`	repeating seasonal component
`events`	optionally, the residual or "events" series

References

Cloern, J.E. and Jassby, A.D. (2010) Patterns and scales of phytoplankton variability in estuarine-coastal ecosystems. *Estuaries and Coasts* **33,** 230–241.

See Also

`decomp_cj`

---

ecometab *Ecosystem metabolism*

Description

Estimate ecosystem metabolism using the Odum open-water method. Estimates of daily integrated gross production, total respiration, and net ecosystem metabolism are returned.

Usage

```
ecometab(dat_in, ...)

## S3 method for class 'swmpr'
ecometab(dat_in, depth_val = NULL, metab_units = "mmol", trace = FALSE, ...)

## Default S3 method:
ecometab(
  dat_in,
  tz,
  lat,
```

```
  long,
  depth_val = NULL,
  metab_units = "mmol",
  trace = FALSE,
  ...
)
```

Arguments

dat_in	Input data object, see details for required time series
...	arguments passed to other methods
depth_val	numeric value for station depth (m) if time series is not available
metab_units	chr indicating desired units of output for oxygen, either as mmol or grams
trace	logical indicating if progress is shown in the console
tz	chr string for timezone, e.g., 'America/Chicago'
lat	numeric for latitude
long	numeric for longitude (negative west of prime meridian)

Details

Input data include both water quality and weather time series, which are typically collected with independent instrument systems. For SWMP data, this requires merging water quality and meteorology swmpr data objects using the comb function (see examples). For the default method not using SWMP data, the input data.frame must have columns named datetimestamp (date/time column, as POSIXct object), do_mgl (dissolved oxygen, mg/L), depth (depth, m), atemp (air temperature, C), sal (salinity, psu), temp (water temperature, C), wspd (wind speed, m/s), and bp (barometric pressure, mb).

The open-water method is a common approach to quantify net ecosystem metabolism using a mass balance equation that describes the change in dissolved oxygen over time from the balance between photosynthetic and respiration processes, corrected using an empirically constrained air-sea gas diffusion model (see Ro and Hunt 2006, Thebault et al. 2008). The diffusion-corrected DO flux estimates are averaged separately over each day and night of the time series. The nighttime average DO flux is used to estimate respiration rates, while the daytime DO flux is used to estimate net primary production. To generate daily integrated rates, respiration rates are assumed constant such that hourly night time DO flux rates are multiplied by 24. Similarly, the daytime DO flux rates are multiplied by the number of daylight hours, which varies with location and time of year, to yield net daytime primary production. Respiration rates are subtracted from daily net production estimates to yield gross production rates. The metabolic day is considered the 24 hour period between sunsets on two adjacent calendar days.

Areal rates for gross production and total respiration are based on volumetric rates normalized to the depth of the water column at the sampling location, which is assumed to be well-mixed, such that the DO sensor is reflecting the integrated processes in the entire water column (including the benthos). Water column depth is calculated as the mean value of the depth variable across the time series in the swmpr object. Depth values are floored at one meter for very shallow stations and 0.5 meters is also added to reflect the practice of placing sensors slightly off of the bottom. A user-supplied depth value can also be passed to the depth_val argument, either as a single value that is

repeated or as a vector equal in length to the number of rows in the input data. An accurate depth value should be used as this acts as a direct scalar on metabolism estimates.

The air-sea gas exchange model is calibrated with wind data either collected at, or adjusted to, wind speed at 10 m above the surface. The metadata should be consulted for exact height. The value can be changed manually using a `height` argument, which is passed to `calckl`.

A minimum of three records are required for both day and night periods to calculate daily metabolism estimates. Occasional missing values for air temperature, barometric pressure, and wind speed are replaced with the climatological means (hourly means by month) for the period of record using adjacent data within the same month as the missing data.

All DO calculations within the function are done using molar units (e.g., mmol O2 m-3). The output can be returned as mass units by setting `metab_units = 'grams'` (i.e., 1mol = 32 g O2, which multiplies all estimates by 32 g mol-1/1000 mg/g). Input data must be in standard mass units for DO (mg L-1).

The specific approach for estimating metabolism with the open-water method is described in Caffrey et al. 2013 and references cited therein.

Value

A `data.frame` of daily integrated metabolism estimates is returned. If a `swmpr` object is passed to the function, this `data.frame` is added to the `metab` attribute and the original object is returned. See the examples for retrieval from a `swmpr` object. The metabolism `data.frame` contains the following:

`date` The metabolic day, defined as the 24 hour period starting at sunrise (calculated using `metab_day`)

`DOF_d` Mean DO flux during day hours, mmol m-2 hr-1. Day hours are calculated using the `metab_day` function.

`D_d` Mean air-sea gas exchange of DO during day hours, mmol m-2 hr-1

`DOF_n` Mean DO flux during night hours, mmol m-2 hr-1

`D_n` Mean air-sea gas exchange of DO during night hours, mmol m-2 hr-1

`Pg` Gross production, mmol m-2 d-1, calculated as ((DOF_d - D_d) - (DOF_n - D_n)) * day hours

`Rt` Total respiration, mmol m-2 d-1, calculated as (DOF_n - D_n) * 24

`NEM` Net ecosytem metabolism, mmol m-2 d-1, calculated as Pg + Rt

References

Caffrey JM, Murrell MC, Amacker KS, Harper J, Phipps S, Woodrey M. 2013. Seasonal and interannual patterns in primary production, respiration and net ecosystem metabolism in 3 estuaries in the northeast Gulf of Mexico. Estuaries and Coasts. 37(1):222-241.

Odum HT. 1956. Primary production in flowing waters. Limnology and Oceanography. 1(2):102-117.

Ro KS, Hunt PG. 2006. A new unified equation for wind-driven surficial oxygen transfer into stationary water bodies. Transactions of the American Society of Agricultural and Biological Engineers. 49(5):1615-1622.

Thebault J, Schraga TS, Cloern JE, Dunlavey EG. 2008. Primary production and carrying capacity of former salt ponds after reconnection to San Francisco Bay. Wetlands. 28(3):841-851.

See Also

calckl for estimating the oxygen mass transfer coefficient used with the air-sea gas exchange model, comb for combining swmpr objects, metab_day for identifying the metabolic day for each observation in the time series, plot_metab for plotting the results, and aggremetab for aggregating the metabolism attribute.

Examples

```
## Not run:
## import water quality and weather data, qaqc
data(apadbwq)
data(apaebmet)
wq <- qaqc(apadbwq)
met <- qaqc(apaebmet)

## combine
dat <- comb(wq, met)

## output units in grams of oxygen
res <- ecometab(dat, metab_units = 'grams')
attr(res, 'metabolism')

## manual input of integration depth
## NA values must be filled
dat_fill <- na.approx(dat, params = 'depth', maxgap = 1e6)
depth <- dat_fill$depth
res <- ecometab(dat, metab_units = 'grams', depth_val = depth)
attr(res, 'metabolism')

## use the default method for ecometab with a generic data frame
## first recreate a generic object from the sample data
cols <- c('datetimestamp', 'do_mgl', 'depth', 'atemp', 'sal', 'temp', 'wspd', 'bp')
dat <- data.frame(dat)
dat <- dat[, cols]
res <- ecometab(dat, tz = 'America/Jamaica', lat = 29.67, long = -85.06)
res

## End(Not run)
```

---

gradcols *Get colors for plots*

Description

Get gradient default colors for plots

Usage

```
gradcols(col_vec = NULL)
```

Arguments

col_vec	chr string of plot colors to use. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.

Details

This is a convenience function for retrieving a color palette. Palettes from RColorBrewer will use the maximum number of colors. The default palette is 'Spectral'.

Value

A character vector of colors in hexidecimal notation.

---

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, ...)
```

Arguments

x	input swmpr object
...	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. The input data must contain only one parameter.

See Also

hist

Examples

```
## get data
data(apadbwq)
dat <- subset(apadbwq, select = 'do_mgl')

## histogram using swmpr method
hist(dat)

## change axis labels, plot title
hist(dat, xlab = 'Dissolved oxygen', main = 'Histogram of DO')

## plot using default 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,
  collMethd = c("1", "2"),
  keep_qaqcstatus = FALSE
)
```

Arguments

`path`	chr string of full path to .csv files with raw data, can be a zipped or unzipped directory where the former must include the .zip extension
`station_code`	chr string of station to import, typically 7 or 8 characters including wq, nut, or met extensions, may include full name with year, excluding file extension
`trace`	logical indicating if progress is sent to console, default `FALSE`
`collMethd`	chr string of nutrient data to subset. 1 indicates monthly, 2 indicates diel. Default is both diel and monthly data.
`keep_qaqcstatus`	logical indicating if the `historical` and `provisionalplus` columns are retained in the output (default `FALSE`), see details

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, specifically those 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. 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 to import files directly from the compressed folder or after the folder is decompressed. In the former case, the requested files are extracted to a temporary directory and then deleted after they are loaded into the current session. An example dataset is available online to illustrate the format of the data provided through the zip downloads feature. See the link below to access these data. All example datasets included with the package were derived from these raw data.

Occasionally, duplicate time stamps are present in the raw data. The function handles duplicate entries differently depending on the data type (water quality, weather, or nutrients). For water quality and nutrient data, duplicate time stamps are simply removed. Note that nutrient data often contain replicate samples with similar but not duplicated time stamps within a few minutes of each other. Replicates with unique time stamps are not removed but can be further processed using `rem_reps`. Weather data prior to 2007 may contain duplicate time stamps at frequencies for 60 (hourly) and 144 (daily) averages, in addition to 15 minute frequencies. Duplicate values that correspond to the smallest value in the frequency column (15 minutes) are retained.

If `keep_qaqcstatus = TRUE`, the `historical` and `provisionalplus` columns are retained in the output. These two columns include integer values as 0 or 1. From the CDMO web page, a value of 0 in the `historical` column indicates that the data have not been through final QAQC by the CDMO. A value of 1 indicates that the data have been through final tertiary review at the CDMO and posted as the final authoritative data. A value of 0 in the `provisionalplus` column indicates that the data have been through the automated flagging process (primary QAQC) only and have not been checked by the Reserve. A value of 1 in the `provisionalplus` column indicates that the data have been through secondary QAQC at the Reserve using Excel macros (provided by the CDMO) to further QAQC the data.

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

Example dataset: https://s3.amazonaws.com/swmpexdata/zip_ex.zip

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`, `rem_reps`, `single_param`

Examples

```
## Not run:
## this is the path for csv example files, decompressed
path <- 'C:/this/is/my/data/path'

## import, do not include file extension
```

```
import_local(path, 'apaebmet')

## this is the path for csv example files, zipped folder
path <- 'C:/this/is/my/data/path.zip'

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

## End(Not run)
```

`map_reserve` *Map a reserve*

Description

Create a map of all the stations in a reserve

Usage

```
map_reserve(nerr_site_id, text_sz = 6, text_col = "black", f = 0.2)
```

Arguments

`nerr_site_id`	chr string of the reserve to map, first three characters used by NERRS or vector of stations to map using the first five characters
`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`
`f`	numeric passed to `make_bbox` as expansion around the bounding box for stations at a NERR site, increase to expand map zoom chr

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 `f` argument may have to be chosen through trial and error depending on the spatial extent of the reserve. A local data file included with the package is used to get the latitude and longitude values of each station. The files includes only active stations as of January 2015.

Value

A `ggplot` object for plotting.

See Also

`get_map`, `ggmap`, `ggplot`

Examples

```
## Not run:
map_reserve('jac')

## End(Not run)
```

`metab_day` *Identify metabolic days in a time series*

Description

Identify metabolic days in a time series based on sunrise and sunset times for a location and date. The metabolic day is considered the 24 hour period between sunsets for two adjacent calendar days. Sunrise and sunset times are estimated using `sun_times`.

Usage

```
metab_day(dat_in, ...)

## Default S3 method:
metab_day(dat_in, tz, lat, long, ...)
```

Arguments

`dat_in`	data.frame
`...`	arguments passed to or from other methods
`tz`	chr string for timezone, e.g., 'America/Chicago'
`lat`	numeric for latitude
`long`	numeric for longitude (negative west of prime meridian)

Details

This function is only used within `ecometab` and should not be called explicitly.

See Also

`ecometab`, `sun_times`

`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

```
data(apadbwq)
dat <- qaqc(apadbwq)
dat <- subset(dat, select = 'do_mgl',
 subset = c('2013-01-22 00:00', '2013-01-26 00:00'))

# interpolate, maxgap of 10 records
```

```
fill1 <- na.approx(dat, params = 'do_mgl', maxgap = 10)

# interpolate maxgap of 30 records
fill2 <- na.approx(dat, params = 'do_mgl', maxgap = 30)

# plot for comparison
par(mfrow = c(3, 1))
plot(fill1, col = 'red', main = 'Interpolation - maximum gap of 10 records')
lines(dat)
plot(fill2, col = 'red', main = 'Interpolation - maximum gap of 30 records')
lines(dat)
```

`overplot` *Plot multiple SWMP time series on the same y-axis*

Description

Plot multiple SWMP time series on the same y-axis, aka overplotting

Usage

```
overplot(dat_in, ...)

## S3 method for class 'swmpr'
overplot(
  dat_in,
  select = NULL,
  subset = NULL,
  operator = NULL,
  ylabs = NULL,
  xlab = NULL,
  cols = NULL,
  lty = NULL,
  lwd = NULL,
  pch = NULL,
  type = NULL,
  ...
)

## Default S3 method:
overplot(
  dat_in,
  date_var,
  select = NULL,
  ylabs = NULL,
  xlab = NULL,
  cols = NULL,
  lty = NULL,
```

```
  lwd = NULL,
  inset = -0.15,
  cex = 1,
  xloc = "top",
  yloc = NULL,
  pch = NULL,
  type = NULL,
  ...
)
```

Arguments

dat_in	input data object
...	additional arguments passed to plot
select	chr string of variable(s) to plot, passed to subset. This is a required argument for the default method.
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.
operator	chr string specifiying binary operator (e.g., '>', '<=') if subset is one date value, passed to subset
ylabs	chr string of labels for y-axes, default taken from select argument
xlab	chr string of label for x-axis
cols	chr string of colors to use for lines
lty	numeric indicating line types, one value for all or values for each parameter
lwd	numeric indicating line widths, one value for all or values for each parameter, used as cex for point size if type = 'p'
pch	numeric for point type of points are used
type	character string indicating 'p' or 'l' for points or lines, as a single value for all parameters or a combined vector equal in length to the number of parameters
date_var	chr string of the name for the datetimestamp column, not required for swmpr objects
inset	numeric of relative location of legend, passed to legend
cex	numeric of scale factor for legend, passed to legend
xloc	x location of legend, passed to legend
yloc	y location of legend, passed to legend

Details

One to many SWMP parameters can be plotted on the same y-axis to facilitate visual comparison. This is commonly known as overplotting. The building blocks of this function include plot, legend, axis, and mtext.

Value

An R plot created using base graphics

See Also

subset

Examples

```
## import data
data(apacpwq)
dat <- qaqc(apacpwq)

## plot
overplot(dat)

## a truly heinous plot
overplot(dat, select = c('depth', 'do_mgl', 'ph', 'turb'),
 subset = c('2013-01-01 0:0', '2013-02-01 0:0'), lwd = 2)

## Not run:
## change the type argument if plotting discrete and continuous data
swmp1 <- apacpnut
swmp2 <- apaebmet
dat <- comb(swmp1, swmp2, timestep = 120, method = 'union')
overplot(dat, select = c('chla_n', 'atemp'), subset = c('2012-01-01 0:0', '2013-01-01 0:0'),
 type = c('p', 'l'))

## End(Not run)
```

---

oxySol *Dissolved oxygen at saturation*

Description

Finds dissolved oxygen concentration in equilibrium with water-saturated air. Function and documentation herein are from archived wq package.

Usage

```
oxySol(t, S, P = NULL)
```

Arguments

t	numeric for temperature, degrees C
S	numeric for salinity, on the Practical Salinity Scale
P	numeric for pressure, atm

Details

Calculations are based on the approach of Benson and Krause (1984), using Green and Carritt's (1967) equation for dependence of water vapor partial pressure on t and S. Equations are valid for temperature in the range 0-40 C and salinity in the range 0-40.

Value

Dissolved oxygen concentration in mg/L at 100% saturation. If `P = NULL`, saturation values at 1 atm are calculated.

References

Benson, B.B. and Krause, D. (1984) The concentration and isotopic fractionation of oxygen dissolved in fresh-water and seawater in equilibrium with the atmosphere. *Limnology and Oceanography* **29,** 620-632.

Green, E.J. and Carritt, D.E. (1967) New tables for oxygen saturation of seawater. *Journal of Marine Research* **25,** 140-147.

`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 web results for swmpr*

Description

Parsing function for objects returned from CDMO web services

Usage

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

Arguments

resp_in	web object returned from CDMO server, response class from httr package
parent_in	chr string of parent nodes to parse

Details

This function parses XML objects returned from the CDMO 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, ...)
```

Arguments

x	input swmpr object
type	chr string for type of plot, default 'l'. See plot.
...	other arguments passed to par, plot.default

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
data(apadbwq)
swmp1 <- 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 default, add lines
plot(dat, type = 'n')
lines(dat, col = 'red')
```

---

plot_metab *Plot ecosystem metabolism for a swmpr object*

Description

Plot gross production, total respiration, and net ecosystem metabolism for a swmpr object.

Usage

```
plot_metab(swmpr_in, ...)

## S3 method for class 'swmpr'
plot_metab(
  swmpr_in,
  by = "months",
  alpha = 0.05,
  width = 10,
  pretty = TRUE,
  ...
)
```

Arguments

swmpr_in	input swmpr object
...	arguments passed to or from other methods
by	chr string describing aggregation period, passed to aggremetab. See details for accepted values.
alpha	numeric indicating alpha level for confidence intervals in aggregated data. Use NULL to remove from the plot.
width	numeric indicating width of top and bottom segments on error bars
pretty	logical indicating use of predefined plot aesthetics

Details

A plot will only be returned if the metabolism attribute for the swmpr object is not NULL. Daily metabolism estimates are also aggregated into weekly averages. Accepted aggregation periods are 'years', 'quarters', 'months', 'weeks', and 'days' (if no aggregation is preferred).

By default, pretty = TRUE will return a ggplot object with predefined aesthetics. Setting pretty = FALSE will return the plot with minimal modifications to the ggplot object. Use the latter approach for easier customization of the plot.

Value

A ggplot object which can be further modified.

See Also

aggremetab, ecometab

Examples

```
## Not run:
## import water quality and weather data
data(apadbwq)
data(apaebmet)

## qaqc, combine
wq <- qaqc(apadbwq)
met <- qaqc(apaebmet)
dat <- comb(wq, met)

## estimate metabolism
res <- ecometab(dat)

## plot
plot_metab(res)

## change alpha, aggregation period, widths
plot_metab(res, by = 'quarters', alpha = 0.1, widths = 0)

## plot daily raw, no aesthetics
```

```
plot_metab(res, by = 'days', pretty = FALSE)

## note the difference if aggregating with a moving window average
plot_metab(res, by = 30)

## End(Not run)
```

---

plot_quants *Create a plot of data for a single year overlaid on historical data.*

Description

A line for a single year is plotted over ribbons ofquantiles for historical data.

Usage

```
plot_quants(swmpr_in, ...)

## S3 method for class 'swmpr'
plot_quants(
  swmpr_in,
  paramtoplot,
  yr,
  yrstart,
  yrend,
  yaxislab = NULL,
  yrcolor = "red3",
  bgcolor1 = "lightgray",
  bgcolor2 = "gray65",
  maintitle = NULL,
  ...
)
```

Arguments

swmpr_in	input swmpr object.
...	additional arguments passed to or from other methods
paramtoplot	chr string of parameter to plot
yr	numeric of year to feature as a line on the plot
yrstart	numeric of year to begin range of comparison data
yrend	numeric of year to end range of comparison data
yaxislab	chr string for y-axis albel. Default is paramtoplot.
yrcolor	chr string of line color for year of interest
bgcolor1	chr string of color for outer 50% of data range
bgcolor2	chr string of color for middle 50% of data range.

`maintitle` chr string of plot title. Default pastes together site name, parameter name, year to feature, and range of years to use for comparison, e.g. 'GNDBHWQ 2017 Daily Average Temp overlaid on 2006-2016 daily averages'.

Details

The plot is based on aggregates of daily average values for the entire time series. Quantiles (min, 25%, 75%, max) for each individual calendar day (01/01, 01/02, ... 12/31) are used to generate a ribbon plot of historical data and the selected year in `yr` is plotted as a line over the ribbon for historical context.

required packages: dplyr, lubridate, ggplot2, tibble

Value

A a ggplot2 object.

Author(s)

Kimberly Cressman, Marcus Beck

Examples

```
# qaqc
dat <- qaqc(apacpwq)

# generate a plot of salinity for 2013 overlaid on 2012-2013 data
plot_quants(dat, 'sal', yr = 2013, yrstart = 2012, yrend = 2013)

# change some of the defaults
plot_quants(dat, 'sal', yr = 2013, yrstart = 2012, yrend = 2013,
 bgcolor1 = 'lightsteelblue2', bgcolor2 = 'lightsteelblue4',
 yaxislab = 'Salinity (psu)')
```

---

plot_summary *Plot graphical summaries of SWMP data*

Description

Plot graphical summaries of SWMP data for individual parameters, including seasonal/annual trends and anomalies

Usage

```
plot_summary(swmpr_in, ...)

## S3 method for class 'swmpr'
plot_summary(
  swmpr_in,
```

```
  param,
  colsleft = c("lightblue", "lightgreen"),
  colsmid = "lightblue",
  colsright = c("lightblue", "lightgreen", "tomato1"),
  base_size = 11,
  years = NULL,
  plt_sep = FALSE,
  sum_out = FALSE,
  fill = c("none", "monoclim", "interp"),
  ...
)
```

Arguments

swmpr_in	input swmpr object
...	additional arguments passed to other methods
param	chr string of variable to plot
colsleft	chr string vector of length two indicating colors for left plots
colsmid	chr string vector of length one indicating colors for middle plots
colsright	chr string vector of length three indicating colors for right plots
base_size	numeric for text size
years	numeric vector of starting and ending years to plot, default all
plt_sep	logical if a list is returned with separate plot elements
sum_out	logical if summary data for the plots is returned
fill	chr string indicating if missing monthly values are left as is ('none', default), replaced by long term monthly averages ('monoclim'), or linearly interpolated using na.approx

Details

This function creates several graphics showing seasonal and annual trends for a given swmp parameter. Plots include monthly distributions, monthly anomalies, and annual anomalies in multiple formats. Anomalies are defined as the difference between the monthly or annual average from the grand mean. Monthly anomalies are in relation to the grand mean for the same month across all years. All data are aggregated for quicker plotting. Nutrient data are based on monthly averages, wheras weather and water quality data are based on daily averages. Cumulative precipitation data are based on the daily maximum.

Individual plots can be obtained if plt_sep = TRUE. Individual plots for elements one through six in the list correspond to those from top left to bottom right in the combined plot.

Summary data for the plots can be obtained if sum_out = TRUE. This returns a list with three data frames with names sum_mo, sum_moyr, and sum_mo. The data frames match the plots as follows: sum_mo for the top left, bottom left, and center plots, sum_moyr for the top right and middle right plots, and sum_yr for the bottom right plot.

Missing values can be filled using the long-term average across years for each month (fill = 'monoclim') or as a linear interpolation between missing values using na.approx (fill = 'interp').

The monthly average works well for long gaps, but may not be an accurate representation of long-term trends, i.e., real averages may differ early vs late in the time series if a trend exists. The linear interpolation option is preferred for small gaps.

Value

A graphics object (Grob) of multiple ggplot objects, otherwise a list of individual ggplot objects if `plt_sep = TRUE` or a list with data frames of the summarized data if `sum_out = TRUE`.

See Also

ggplot

Examples

```
## import data
data(apacpnut)
dat <- qaqc(apacpnut)

## plot
plot_summary(dat, param = 'chla_n', years = c(2007, 2013))

## get individaul plots
plots <- plot_summary(dat, param = 'chla_n', years = c(2007, 2013), plt_sep = TRUE)

plots[[1]] # top left
plots[[3]] # middle
plots[[6]] # bottom right

## get summary data
plot_summary(dat, param = 'chla_n', year = c(2007, 2013), sum_out = TRUE)
```

---

plot_wind *Create a wind rose*

Description

Create a wind rose from met data

Usage

```
plot_wind(swmpr_in, ...)

## S3 method for class 'swmpr'
plot_wind(
  swmpr_in,
  years = NULL,
  angle = 45,
```

```
  width = 1.5,
  breaks = 5,
  paddle = FALSE,
  grid.line = 10,
  max.freq = 30,
  cols = "GnBu",
  annotate = FALSE,
  main = NULL,
  type = "default",
  between = list(x = 1, y = 1),
  par.settings = NULL,
  strip = NULL,
  ...
)
```

Arguments

swmpr_in	input swmpr object
...	arguments passed to or from other methods
years	numeric of years to plot, defaults to most recent
angle	numeric for the number of degrees occupied by each spoke
width	numeric for width of paddles if paddle = TRUE
breaks	numeric for the number of break points in the wind speed
paddle	logical for paddles at the ends of the spokes
grid.line	numeric for grid line interval to use
max.freq	numeric for the scaling used to set the maximum value of the radial limits (like zoom)
cols	chr string for colors to use for plotting, can be any palette R recognizes or a collection of colors as a vector
annotate	logical indicating if text is shown on the bottom of the plot for the percentage of observations as 'calm' and mean values
main	chr string for plot title, defaults to station name and year plotted
type	chr string for temporal divisions of the plot, defaults to whole year. See details.
between	list for lattice plot options, defines spacing between plots
par.settings	list for optional plot formatting passed to lattice.options
strip	list for optional strip formatting passed to strip.custom

Details

This function is a convenience wrapper to windRose. Most of the arguments are taken directly from this function.

The type argument can be used for temporal divisions of the plot. Options include the entire year (type = "default"), seasons (type = "season"), months (type = "month"), or weekdays (type = "weekday"). Combinations are also possible (see windRose).

Value

A wind rose plot

Author(s)

Kimberly Cressman, Marcus Beck

Examples

```
plot_wind(apaebmet)
```

---

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	character string of qaqc flags to keep, default '0', any number of flag codes can be supplied including three character error codes (see examples)
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: https://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. Additionally, simple filters are used to remove obviously bad values, e.g., wind speed values less than zero or pH values greater than 12. Erroneous data entered as -99 are also removed. Processed data will have QAQC columns removed, in addition to removal of values in the actual parameter columns that do not meet the criteria.

The data are filtered by matching the flag columns with the character string provided by qaqc_keep. A single combined string is created by pasting each element together using the 'l' operator, then

using partial string matching with grepl to compare the actual flags in the QAQC columns. Values that can be passed to the function are those described online: https://cdmo.baruch.sc.edu/data/qaqc.cfm.

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

```
## Not run:
## get data
data(apadbwq)
dat <- apadbwq

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

## retain observations with the 'CSM' error code
qaqc(dat, qaqc_keep = 'CSM')

## End(Not run)
```

---

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: https://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
data(apadbwq)
dat <- apadbwq

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

---

rem_reps *Remove replicates in nutrient data*

Description

Remove replicates in SWMP nutrient data to keep approximate monthly time step

Usage

```
rem_reps(swmpr_in, ...)

## S3 method for class 'swmpr'
rem_reps(swmpr_in, FUN = function(x) mean(x, na.rm = TRUE), ...)
```

Arguments

swmpr_in	input swmpr object
...	arguments passed to other methods
FUN	function to combine values, defaults to mean

Details

Raw nutrient data obtained from the CDMO will usually include replicate samples that are taken within a few minutes of each other. This function combines nutrient data that occur on the same day. The `datetimestamp` column will always be averaged for replicates, but the actual observations will be combined based on the user-supplied function which defauls to the mean. Other suggested functions include the `median`, `min`, or `max`. The entire function call including treatment of `NA` values should be passed to the `FUN` argument (see the examples). The function is meant to be used after `qaqc` processing, although it works with a warning if QAQC columns are present.

Value

Returns a swmpr object for nutrient data with no replicates.

See Also

`qaqc`

Examples

```
## get nutrient data
data(apacpnut)
swmp1 <- apacpnut

# remove replicate nutrient data
rem_reps(swmp1)

# use different function to aggregate replicates
func <- function(x) max(x, na.rm = TRUE)
rem_reps(swmp1, FUN = func)
```

---

setstep *Format a swmpr time vector*

Description

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

Usage

```
setstep(dat_in, ...)

## S3 method for class 'swmpr'
setstep(dat_in, timestep = 15, differ = NULL, ...)

## Default S3 method:
setstep(dat_in, date_col, timestep = 15, differ = NULL, ...)
```

Arguments

dat_in	input data object
...	arguments passed to or from other methods
timestep	numeric value of time step to use in minutes. Alternatively, a chr string indicating 'years', 'quarters', 'months', 'days', or 'hours' can also be used. A character input assumes 365 days in a year and 31 days in a month.
differ	numeric value defining buffer for merging time stamps to standardized time series, defaults to one half of the timestep
date_col	chr string for the name of the date/time column, e.g., "POSIXct" or "POSIXlt" objects

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 of the function, 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 data object for the specified time step

See Also

comb

Examples

```
## Not run:
## import data
data(apaebmet)
dat <- 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
data(apacpnut)
dat_nut <- apacpnut
dat_nut <- setstep(dat_nut, timestep = 60)
subset(dat_nut, rem_rows = TRUE, rem_cols = TRUE)
```

```
## End(Not run)
```

`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, trace = TRUE)
```

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
`trace`	logical indicating if import progress is printed in console

Details

This function retrieves data from the CDMO through the web services URL. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: https://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 retrieves data from the CDMO web services. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: https://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 retrieves data from the CDMO web services. The computer making the request must have a registered IP address. Visit the CDMO web services page for more information: https://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(x, ...)

## Default S3 method:
smoother(x, window = 5, sides = 2, ...)

## S3 method for class 'swmpr'
smoother(x, params = NULL, ...)
```

Arguments

x	input object
...	arguments passed to or from other methods
window	numeric vector defining size of the 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
data(apadbwq)
swmp1 <- 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)
```

---

stat_locs *Locations of NERRS sites*

Description

Location of NERRS sites in decimal degrees and time offset from Greenwich mean time. Only active sites are included. Sites are identified by five letters indexing the reserve and site names. The dataset is used to plot locations with the `map_reserve` function and to identify metabolic days with the `ecometab` function. Created from sampling_locations.csv provided by CDMO.

Usage

```
stat_locs
```

Format

A data.frame object with 161 rows and 4 variables:

station_code chr

latitude numeric

longitude numeric

gmt_off int

See Also

ecometab, map_reserve

---

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, 'datetimestamp' will always be kept
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., `%Y-%m-%d %H:%M` in POSIX standards). Be aware that an error may be returned using this function if the subset argument is in the correct format but the calendar date does not exist, e.g. `'2012-11-31 12:00'`. The function can also be used to remove rows and columns that do not contain data, which may be useful after processing with other functions.

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
data(apaebmet)
dat <- apaebmet

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

## 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'))
```

---

sun_times *Estimate sunrise and sunset times*

Description

Estimate sunrise and sunset times for a location and date. This is a stripped down version of `getSunlightTimes` from the suncalc package, which implements the SunCalc algorithm by Vladimir Agafonkin (https://github.com/mourner/suncalc).

Usage

```
sun_times(date, lat, long, tz = "UTC")
```

Arguments

date	Date vector, or a vector that can be converted with as.Date
lat	numeric for latitude
long	numeric for longitude (negative west of prime meridian)
tz	chr string for timezone of the output, e.g., 'America/Chicago'

Details

Times are estimated for a solar altitude of -0.833 degrees, which accounts for atmospheric refraction and the radius of the solar disc. Estimates are typically within one to two minutes of those from the NOAA solar calculator for mid-latitude locations. Times for each date are based on the solar transit nearest to noon UTC on that date.

This function is used within metab_day to identify metabolic days for ecometab.

Value

A data.frame with columns date, sunrise, and sunset. The latter two are POSIXct in the timezone given by tz. Values are NA if the sun does not rise or set on a date, e.g., polar day or night.

See Also

metab_day

Examples

```
# Apalachicola Bay NERR, Florida
sun_times(as.Date('2024-06-21'), lat = 29.67, long = -85.06, tz = 'America/Jamaica')

# a sequence of dates
dts <- seq(as.Date('2024-01-01'), as.Date('2024-01-07'), by = 'days')
sun_times(dts, lat = 29.67, long = -85.06, tz = 'America/Jamaica')
```

---

swmpr *Create a swmpr 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. Attributes of a swmpr object include `names`, `row.names`, `class`, `station`, `parameters`, `qaqc_cols`, `cens_cols`, `date_rng`, `timezone`, `stamp_class`, `metabolism` (if present), and `metab_units` (if present).

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

∗ **analyze**
aggremetab, 3
aggreswmp, 4
decomp, 16
decomp_cj, 19
ecometab, 22
hist.swmpr, 26
map_reserve, 29
na.approx.swmpr, 31
overplot, 32
plot.swmpr, 36
plot_metab, 37
plot_quants, 39
plot_summary, 40
plot_wind, 42
smoother, 51
∗ **datasets**
apacpnut, 8
apacpwq, 9
apadbwq, 10
apaebmet, 11
stat_locs, 52
∗ **organize**
cens_id, 14
comb, 15
qaqc, 44
qaqcchk, 45
rem_reps, 46
setstep, 47
subset.swmpr, 53
∗ **retrieve**
all_params, 6
all_params_dtrng, 7
import_local, 27
single_param, 49
site_codes, 50
site_codes_ind, 50

aggregate, *3*, *5*
aggremetab, 3, *25*, *38*
aggreswmp, *3*, 4
all_params, 6, *7*, *28*, *49*
all_params_dtrng, *6*, 7, *7*, *28*, *49*
apacpnut, 8
apacpwq, 9
apadbwq, 10
apaebmet, 11
as.Date, *55*
axis, *33*

calckl, 13, *24, 25*
cens_id, 14
comb, 15, *23*, *25*, *48*

data.frame, *3*, *5*, *8–10*, *12*, *46*, *50*, *53*
decomp, 16
decomp.swmpr, *20*
decomp_cj, 19, *22*
decompose, *16–18*, *20*
decompTs, 21

ecometab, *3*, *13*, 22, *30*, *38*, *52, 53*, *55*

filter, *52*

geom_text, *29*
get_map, *29*
ggmap, *29*
ggplot, *5*, *20*, *29*, *38*, *42*
ggplot2, *40*
gradcols, 25
grepl, *45*

hist, *26*
hist.swmpr, 26

IDate, *5*
import_local, *8–11*, 27, *36*

lattice.options, *43*
legend, *33*

lines.swmpr *(plot.swmpr)*, 36

make_bbox, *29*
map_reserve, 29, *52, 53*
max, *47*
mean, *3*, *5*
median, *47*
metab_day, *24, 25*, 30, *55*
min, *47*
mtext, *33*

na.approx, *31*, *41*
na.approx.swmpr, 31

overplot, 32
oxySol, 34

param_names, 35
parser, 36
plot, *33*, *36, 37*
plot.swmpr, 36
plot_metab, *3*, *25*, 37
plot_quants, 39
plot_summary, 40
plot_wind, 42
POSIXct, *23*, *56*

qaqc, *14, 15*, 44, *46, 47*
qaqcchk, 45, *45*

rem_reps, *28*, 46

setstep, *16*, *18*, 47
single_param, *7*, *28*, 49
site_codes, 50
site_codes_ind, 50
smoother, *3*, 51
stat_locs, 52
stl, *18*
strip.custom, *43*
structure, *56*
subset, *33, 34*, *54*
subset.swmpr, 53
sun_times, *30*, 54
swmpr, *8–12*, *23, 24*, *33*, *36*, *38*, 55

time_vec, 56
ts, *16–18*, *20*

windRose, *43*