SWMPr: An R package for retrieving, organizing, and analyzing environmental data for estuaries

Marcus William Beck[1, *]

1 ORISE Research Participation Program, USEPA National Health and Environmental Effects Research Laboratory, Gulf Ecology Division, 1 Sabine Island Drive, Gulf Breeze, FL 32651, USA

*** beck.marcus@epa.gov**

Abstract

Standardized monitoring programs have vastly improved the quantity and quality of data that form the basis of environmental decision-making. One example in the United States is the System Wide Monitoring Program (SWMP) that was implemented in 1995 by the federally-funded National Estuarine Research Reserve System (NERRS). This program has provided two decades of continuous monitoring data at over 300 fixed stations in 28 estuaries across the United States. SWMP data have been used in a variety applications with the general objective of describing dynamics of estuarine ecosystems to better inform effective coastal management. However, simple tools for processing and evaluating the large and increasing quantity of data provided by the monitoring network have prevented large-scale comparisons between systems and, in some cases, simple trend analysis of water quality parameters at individual sites. We describe a new open-source software package, SWMPr, developed in program R for use with SWMP environmental data. The package provides several functions that facilitate data retrieval, organization, and analysis of time series data to describe water quality, weather, and nutrient dynamics in the reserve estuaries. Previously unavailable functions for estuaries are also provided to estimate rates of ecosystem metabolism using the open-water method. Tools included with the SWMPr package have facilitated a cross-reserve comparison of trends, including simple evaluation of changes over time and comparisons of patterns in primary productivity. Overall, the package provides an effective approach to link quantitative information with analysis tools that will greatly inform management programs aimed at coastal protection and restoration.

Introduction

The development of low-cost, automated sensors that have the ability to collect data in near real-time has enabled a proliferation of standardized environmental monitoring programs [1, 2]. These programs provide access to invaluable sources of data that can be used to address a variety of research and management objectives. Applications from automated remote sensors are numerous with notable examples including prediction of harmful algal blooms and toxicants in aquatic systems [3], development of a hydrometeorological monitoring network to support flash flood warning programs [4], and automated detection of airborne chemical warfare agents [5]. Automated remote monitoring programs offer several advantages over traditional site-specific, field-based

methods including streamlining of data acquisition, minimizing human error, and reducing the overall cost of the collection process [1]. However, the growing quantity of available information to address relevant questions has contributed to the notion of 'big data' science where analysis limitations are more often defined by compututational requirements and signal identification in the presence of noise rather than the availability of information. Growing concerns over the use of adequate quality assurance and control methods, challenges for synthesis and interpretation, and increased emphasis on exploratory-based analytical techniques have characterized applications of data from automated monitoring programs [6,7].

The National Estuarine Research Reserve System (NERRS, http://www.nerrs.noaa.gov/) is a network of 28 estuarine reserves in the United States that was established by the Coastal Zone Management Act of 1972. The reserves represent different biogeographic regions and estuarine types that were chosen to address multiple goals for long-term research, monitoring, education, and stewardship in support of coastal management. As part of this effort, the System Wide Monitoring Program (SWMP) program was implemented in 1995 at over 300 stations at each of the reserves to provide a robust, long-term monitoring system for water quality, weather, and land-use/habitat change. The SWMP network has provided a continous source of data collected at near real-time at each of the reserves with the intent to evaluate natural and anthrogogenic causes of spatiotemporal variation in environmental condition and ecosystem function. These data have been useful for evaluating relevant characteristics at individual reserves (eg., [8,9]) and differences between reserves (e.g., ecosystem metabolism [10,11], tidal characteristics [12], dissolved oxygen [13]). However, no cross-reserve comparisons have been conducted within the last decade despite the online availability of current SWMP data. NERRS researchers and staff have also expressed a need for quantitative analysis tools to evaluate trends in water quality time series given the quantity of data provided by SWMP [14].

This article describes a software package that was developed to address research needs of the NERRS program using the open-source statistical programming language R [15]. SWMPr (pronounced 'swamper') is an R package that contains functions for retrieving, organizing, and analyzing estuary monitoring data from the System Wide Monitoring Program (SWMP). Functions provided by SWMPr address many of the common issues working with large datasets created from automated sensor networks, such as data pre-processing to remove unwanted information, combining data from different sources, and exploratory analyses to identify key parameters of interest. Additionally, a cross-reserve comparison of current ecosystem metabolism estimates is provided to illustrate a potential application using the functions in this package. The software is provided specifically for use with NERRS data, although many of the applications are relevant addressing common challenges working with large datasets.

SWMP overview and data retrieval

Four core data elements are collected through the SWMP monitoring network: abiotic monitoring data, biotic observations, habitat and land use mapping, and sentinel monitoring. The SWMPr package is developed for the continous abiotic monitoring network which includes a majority of the data collected by SWMP. Abiotic elements monitored at each reserve include water quality (water temperature, specific conductivity, salinity, dissolved oxygen concentration, dissolved oxygen saturation, depth, pH, turbidity, chlorophyll fluorescence), weather (air temperature, relative humidity, barometric pressure, wind speed, wind direction, photosynthetically active radiation, precipitation), and nutrient data (orthophosphate, ammonium, nitrite, nitrate, nitrite + nitrate, chlorophyll a). Each reserve has no less than four water

quality stations and one weather station at fixed locations. Water quality and weather data are collected at 15 minute intervals, whereas nutrient data are collected monthly at each water quality station. All data are made accessible through the Centralized Data Management Office (CDMO) web portal, where multiple quality assurance/quality control (QAQC) measures are used to screen the information. The final data include all observations with relevant QAQC flags indicating the appropriate qualifier (view codes).

The CDMO web portal was established to support priority areas of SWMP that focus on the continuation and advancement of data and information management. As such, CDMO provides access to over 35 million water quality, weather, and nutrient records that have been authenticated through systematic QAQC procedures. Estuary data must be obtained from CDMO prior to using most of the functions within the SWMPr package. In most cases, the analysis needs will typically define the location, date range, and parameters of interest that need to be obtained from CDMO. All stations in the SWMP network are identified by a 7 or 8 character name that specifies the reserve, station, and parameter type. For example, 'apaebwq' is the water quality identifier for the East Bay station at the Apalachicola reserve. Similarly, a suffix of 'met' or 'nut' would specify the weather (meteorological) or nutrients station.

SWMP data can be used in R after they are obtained directly from the CDMO through an online query or by using the package's retrieval functions. Prior to any data request, the site, parameter type, and date ranges need to be identified based on the analysis needs. All reserve names, stations, and date ranges for the water quality, weather, or nutrients data can be viewed on the CDMO website. Alternatively, the `site_codes` (all sites) or `site_codes_ind` (single site) functions provided by the SWMPr package can be used to view the same information. As noted below, the computer's IP address must be registered by CDMO staff before using the data retrieval functions in SWMPr. Web services are provided by CDMO to provide direct access to SWMP data through http requests, in addition to standard graphical user interface options for selecting data. The data retrieval functions in SWMPr are simple calls to the existing retrieval functions on CDMO web services. For example, the `all_params` function in SWMPr uses the `exportAllParamsXMLNew` function from the web services to retrieve metadata for all the SWMP sites. The text below describes the data retrieval functions in more detail.

Structure of the SWMPr package

Installing the package

The SWMPr package was developed for use with the R statistical programming language and a recent version of R (v3.0.0 or greater) should be installed (see http://cran.r-project.org/). The SWMPr package can be installed from GitHub by executing the following commands at the R terminal. The package is loaded in the current workspace by using the `library` command.

```
install.packages('devtools')
library(devtools)
install_github('fawda123/SWMPr')
library(SWMPr)
```

The SWMPr package was developed by considering a standard workflow that categorizes the functions as one of three steps based on their intended use: *retrieving*, *organizing*, and *analyzing*. Functions for retrieving are used to import the data into R as a `swmpr` object class. Functions for organizing and analyzing the data provide methods

Table 1. Retrieval functions available from the SWMPr package. Full documentation for each function is in the help file (e.g., execute `?all_params` at the command line).

Function	Description
`all_params`	Retrieve up to 100 records starting with the most recent at a given station, all parameters. Wrapper to `exportAllParamsXMLNew` function on web services.
`all_params_dtrng`	Retrieve records of all parameters within a given date range for a station. Optional argument for a single parameter. Maximum of 1000 records. Wrapper to `exportAllParamsDateRangeXMLNew`.
`import_local`	Import files from a local path. The files must be in a specific format, specifically those returned from the CDMO using the zip downloads option for a reserve.
`single_param`	Retrieve up to 100 records for a single parameter starting with the most recent at a given station. Wrapper to `exportSingleParamXMLNew` function on web services.
`site_codes`	Metadata for all stations, wrapper to `exportStationCodesXMLNew` function on web services.
`site_codes_ind`	Metadata for all stations at a single site, wrapper to `NERRFilterStationCodesXMLNew` function on web services.

for working with the `swmpr` object class. An additional class of functions, termed 'miscellaneous', are also included as helpers for the main functions. The following describes a general approach for using each category of functions based on a standard data workflow.

Data retrieval

SWMP data must be obtained from the CDMO prior to organizing and analyzing the information. Two approaches can be used. First, the data can be obtained outside of R using the graphical query forms on the CDMO website (see here). Second, functions from the SWMPr package can be used to import the data directly from the online server using CDMO web services (Table 1). In the latter case, the IP address for the computer making the request must be registered with CDMO. This can be done by following instructions here. The `site_codes` or `site_codes_ind` functions can be used to view the available metadata after a computer is registered with CDMO.

```
# retrieve metadata for all sites
site_codes()

# retrieve metadata for a single site
site_codes_ind('apa')
```

Due to rate limitations on the CDMO server, the retrieval functions return a limited number of records. The functions are more useful for evaluating short time periods, although these functions could be used iteratively (i.e., with `for` loops) to obtain longer time series. Data retrieval functions to access the CDMO include `all_params`, `all_params_dtrng`, and `single_param`. These are functions that call the existing web protocol methods on the CDMO web services. `all_params` returns the most recent

records of all parameters at a station, `all_params_dtrng` returns all records within a date range for all parameters or a single parameter, and `single_param` is identical to `all_params` except that a single parameter is requested.

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

# 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')

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

For larger requests, data are easier to obtain outside of R using the CDMO query system and then importing using the `import_local` function. Data can be retrieved from the CDMO several ways. The `import_local` 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. The zip downloads feature is an easy way to obtain data from multiple stations in one request. 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.

Occasionally, duplicate time stamps are present in the raw data. The `import_local` 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.

```
# import data for apaebmet that you downloaded

# this is an example path with the csv files, change as needed
path <- 'C:/my_path/'

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

All data retrieval functions return a `swmpr` object that includes relevant data and several attributes describing the dataset. The data include a `datetimestamp` column in the appropriate timezone for a station. Note that the `datetimestamp` is standard time for each timezone and does not include daylight savings. Additional columns include parameters for a given data type (weather, nutrients, or water quality) and correspondingg QAQC columns if returned from the initial data request. The attributes for a `swmpr` object include `names` of the dataset, `row.names` of the dataset, `class` (character string indicating `swmpr` and `data.frame`) `station` (7 or 8 characters

identifying the station), **parameters** (character vector of data columns, e.g., **'do_mgl'**), **qaqc_cols** (logical **T** or **F** if present or not), **date_rng** (**POSIXct** vector of minimum/maximum dates), **timezone** (text string in country/city format), and **stamp_class** (class of **datetimestamp** vector, **POSIXct** or **Date**). Attributes of a **swmpr** object can be viewed as follows.

```
# import binary data
data(apadbwq)
dat <- apadbwq

# verify that dat is swmpr class
class(dat)

## [1] "swmpr"      "data.frame"

# all attributes of dat
names(attributes(dat))

## [1] "names"       "row.names"   "class"       "station"
## [5] "parameters"  "qaqc_cols"   "date_rng"    "timezone"
## [9] "stamp_class"

# a single attribute of dat
attr(dat, 'station')

## [1] "apadbwq"
```

The **swmpr** object class was created for use with specific methods following the S3 object definition approach [16]. A **swmpr** object also secondarily inherits methods from the **data.frame** class, such that common **data.frame** methods also apply to **swmpr** objects. Available methods for the **swmpr** class are described below and can also be viewed:

```
# available methods for swmpr class
methods(class = 'swmpr')

##  [1] aggregate.swmpr       aggregate_metab.swmpr comb.swmpr
##  [4] decomp.swmpr          decomp_cj.swmpr       ecometab.swmpr
##  [7] hist.swmpr            lines.swmpr           na.approx.swmpr
## [10] plot.swmpr            plot_metab.swmpr      plot_summary.swmpr
## [13] qaqc.swmpr            qaqcchk.swmpr         rem_reps.swmpr
## [16] setstep.swmpr         smoother.swmpr        subset.swmpr
```

Example data as raw, comma-separated files have not been included in the package due to size limitations. However, a sample dataset can be downloaded for use with the examples below. This dataset has an identical format as the data returned from the zip downloads feature of the CDMO. Processed verions of the raw data are included with the package as binary data files (.RData) to decrease processing times with the examples. Information for each binary file can be viewed as follows.

```
# view help files for complementary data
```

```
# all files are samples from Apalachicola Bay

# cat point station, nutrients
?apacpnut

# cat point station, water quality
?apacpwq

# dry bar station, water quality
?apadbwq

# east bay station, weater
?apaebmet
```

Data organizing

The retrieval functions import the data into R as a swmpr object for use with the organize and analyze functions. The organize functions are used to clean or prepare the data for analysis, including removal of QAQC flags, subsetting, creating a standardized time series vector, and combining data of different types (Table 2).

The `qaqc` function is a simple screen to retain values from the data with specified QAQC flags (described here). 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 `QAQC` 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.

```
# 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))
```

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 with), `qaqc`. The `qaqcchk` function can be used to view this information. Consult the online documentation for a description of each QAQC flag.

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

Raw nutrient data obtained from the CDMO will usually include replicate samples that were taken within a few minutes of each other. The `rem_reps.swmpr` function combines nutrient data that occur on the same day to preserve an approximate monthly

Table 2. Organizing functions available from the SWMPr package. Full documentation for each function is in the help file (e.g., execute `?comb.swmpr` at the command line).

Function	Description
`comb.swmpr`	Combines `swmpr` objects to a common time series using setstep, such as combining the weather, nutrients, and water quality data for a single station. Only different data types can be combined.
`qaqc.swmpr`	Remove QAQC columns and remove data based on QAQC flag values for a `swmpr` object. Only applies if QAQC columns are present.
`qaqcchk.swmpr`	View a summary of the number of observations in a `swmpr` object that are assigned to different QAQC flags used by CDMO. The output is used to inform further processing but is not used explicitly.
`rem_reps.swmpr`	Remove replicate nutrient data that occur on the same day. The default is to average replicates.
`setstep.swmpr`	Format data from a `swmpr` object to a continuous time series at a given timestep. The function is used in `comb.swmpr` and can also be used with individual stations.
`subset.swmpr`	Subset by dates and/or columns for a `swmpr` object. This is a method passed to the generic `subset` function provided in the base package.

time step. 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.

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

# remove replicate nutrient data
rem_reps(swmp1)

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

A subset method added to the existing `subset` function is available for `swmpr` objects. This function is used to subset the 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`. Finally, the function can be used to remove rows and columns that do not contain data.

```
# 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 = T, rem_cols = T)
```

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 `setstep` function can be used with `subset` and to create a time series at a set interval with empty data removed.

```
# 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 = T, rem_cols = T)
```

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.

```
# get nuts, wq, and met data as separate objects for the same station
# note that most sites usually have one weather station
data(apacpnut)
data(apacpwq)
data(apaebmet)
swmp1 <- apacpnut
swmp2 <- apacpwq
swmp3 <- 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')
```

Data analysis

The analysis functions range from general purpose tools for time series analysis to more specific functions for working with continuous monitoring data in estuaries (Table 3). The latter category includes a limited number of functions that were developed by myself or others. The general purpose tools are `swmpr` methods that were developed for existing generic functions in the R base installation or relevant packages. These functions include `swmpr` methods for `aggregate`, `filter`, and `approx` to deal with missing or noisy data and more general functions for exploratory data analysis, such as `plot`, `summary`, and `hist` methods. Decomposition functions `decomp` and `decomp_cj`) are provided as relatively simple approaches for decomposing time series into additive or multiplicative components. The analysis functions may or may not return a `swmpr` object depending on whether further processing with `swmpr` methods is possible from the output.

The `aggregate` function aggregates parameter data for a `swmpr` object by set periods of observation. This function 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 a user-defined 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 data.table package. This approach was used to facilitate plotting using predefined methods for Date and POSIX objects. Additionally, the method of treating NA values for the aggregation function should be noted since this may greatly affect the quantity of data that are returned (see the example below). Finally, the default argument for `na.action` is set to `na.pass` for `swmpr` objects to preserve the time series of the input data.

Table 3. Analysis functions available from the SWMPr package. Full documentation for each function is in the help file (e.g., execute `?aggregate.swmpr` at the command line).

Function	Description
`aggregate.swmpr`	Aggregate `swmpr` objects for different time periods - years, quarters, months, weeks, days, or hours. Aggregation function is user-supplied but defaults to mean.
`aggregate_metab.swmpr`	Aggregate metabolism data from a `swmpr` object. This is primarly used within `plot_metab` but may be useful for simple summaries of raw daily data.
`ecometab.swmpr`	Estimate ecosystem metabolism for a combined water quality and weatehr dataset using the open-water method.
`decomp.swmpr`	Decompose a `swmpr` time series into trend, seasonal, and residual components. This is a simple wrapper to `decompose`. Decomposition of monthly or daily trends is possible.
`decomp_cj.swmpr`	Decompose a `swmpr` time series into grandmean, annual, seasonal, and events components. This is a simple wrapper to `decompTs` in the wq package. Only monthly decomposition is possible.
`hist.swmpr`	Plot a histogram for a `swmpr` object.
`lines.swmpr`	Add lines to an existing `swmpr` plot.
`na.approx.swmpr`	Linearly interpolate missing data (`NA` values) in a `swmpr` object. The maximum gap size that is interpolated is defined as a maximum number of records with missing data.
`plot.swmpr`	Plot a univariate time series for a `swmpr` object. The parameter name must be specified.
`plot_metab.swmpr`	Plot ecosystem metabolism estimates after running `ecometab` on a `swmpr` object.
`plot_summary.swmpr`	Create summary plots of seasonal/annual trends and anomalies for a water quality or weather parameter.
`smoother.swmpr`	Smooth `swmpr` objects with a moving window average. Window size and sides can be specified, passed to `filter`.

```
# combine, qaqc, remove empty columns
dat <- comb(swmp1, swmp2, method = 'union')
dat <- qaqc(dat)
swmpr_in <- subset(dat, rem_cols = T)

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

# get mean DO by quarters, remove NA when calculating means
fun_in <- function(x) mean(x, na.rm = T)
aggregate(swmpr_in, FUN = fun_in, 'quarters', params = c('do_mgl'))
```

Time series can be smoothed to better characterize a signal independent of noise (Fig. 1). Although there are many approaches to smoothing, a moving window average is intuitive and commonly used. 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. As before, the `params` argument specifies which parameters to smooth. See Fig. 1 for the output from the code.

```
# 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)
```

A common issue with any statistical analysis is the treatment of missing values. Missing data can be excluded from the analysis, included but treated as true zeroes, or interpolated based on similar values. In either case, an analyst should have a strong rationale for the chosen method. A common approach used to handle 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 challenge with linear interpolation is an appropriate gap size for fitting missing observations. The ability of the interpolated data to approximate actual 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 are more likely to resemble actual patterns. An appropriate gap size limit depends on the unique characteristics of specific datasets or parameters. The `na.approx` function can be used to interpolate gaps in a `swmpr` object. A required argument for the function is `maxgap`

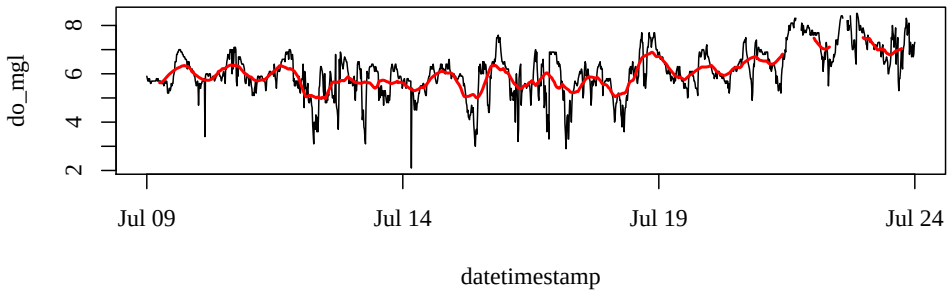

Fig. 1. Raw and smoothed dissolved oxygen data for a two-week period after using the `smoother` function.

which defines the maximum gap size for interpolation. See Fig. 2 for the output from the following code.

```
# get data
data(apadbwq)
swmp1 <- 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 = 'Interpolation - 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')
```

The `decomp` function is a simple wrapper to `decompose` that 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 the help documentation for `decompose`).

The `decompose` function requires a `ts` object with a specified frequency as input.

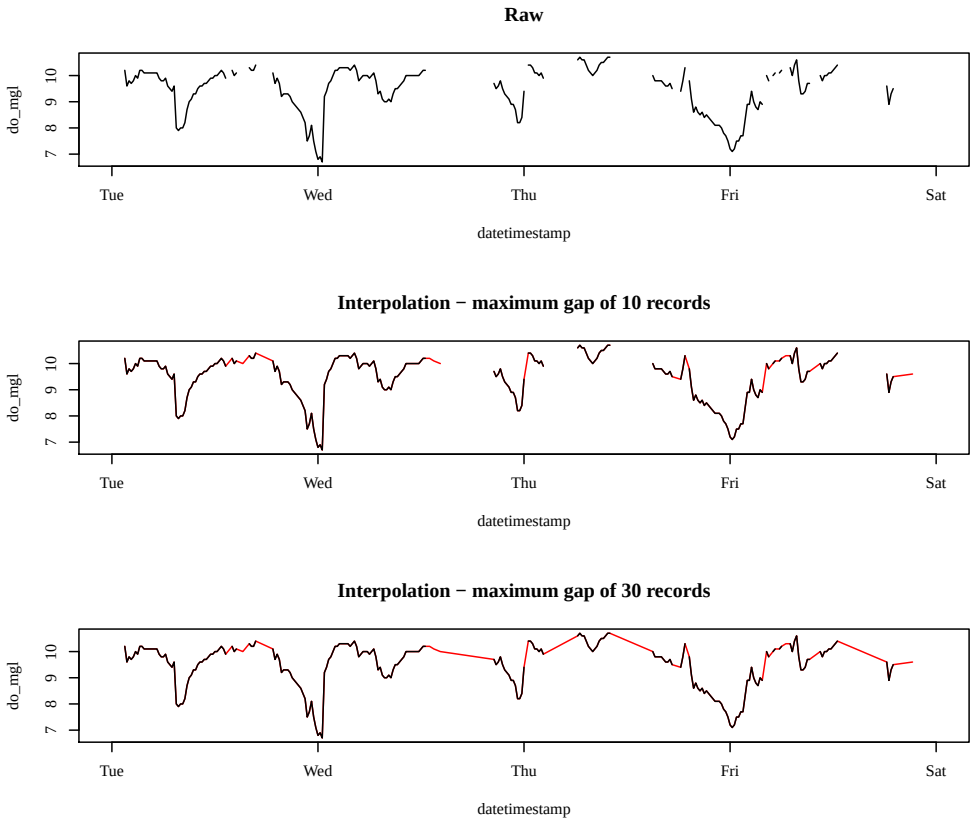

Fig. 2. Examples illustrating use of the `na.approx` function to fill gaps of different sizes in a dissolved oxygen time series for a four day period.

The `decomp` function converts the input `swmpr` vector to a `ts` object prior to `decompose`. This requires an explicit input defining the frequency of the parameter in the time series. 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, character 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 also 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. Fig. 3 is an example of the `decomp` function.

```
# 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)
```

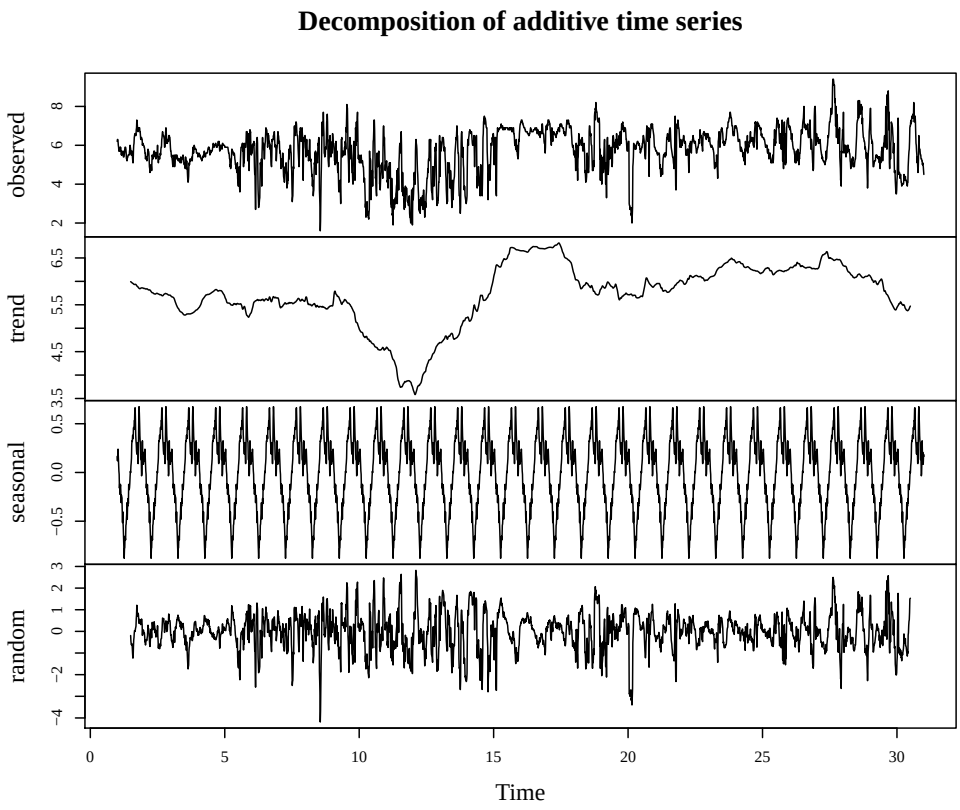

Fig. 3. An additive decomposition of dissolved oxygen into a trend, seasonal, and random component using the `decomp` function.

An alternative approach to time series decomposition is provided by the `decomp_cj` function, which is a simple wrapper to the `decompTs` function in the wq package. Theory describing this method is provided by Cloern and Jassby [17]. The function is similar to `decomp.swmpr` with a few key differences. The `decomp.swmpr` function decomposes the time series into a trend, seasonal, and random component, 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 do not 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 works best for nutrient data that are typically obtained on a monthly cycle. The function will also work with continuous water quality or weather data but note that the data must first aggregated on the monthly scale before decomposition. Additional arguments passed to `decompTs` can be used with `decomp_cj`, such as `startyr`, `endyr`, and `type`. Values passed to `type` are `mult` (default) or `add`, referring to multiplicative or additive decomposition. Fig. 4 shows the results from the `decomp_cj` function applied to a multi-year chlorophyll time series.

```
# get data
```

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

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

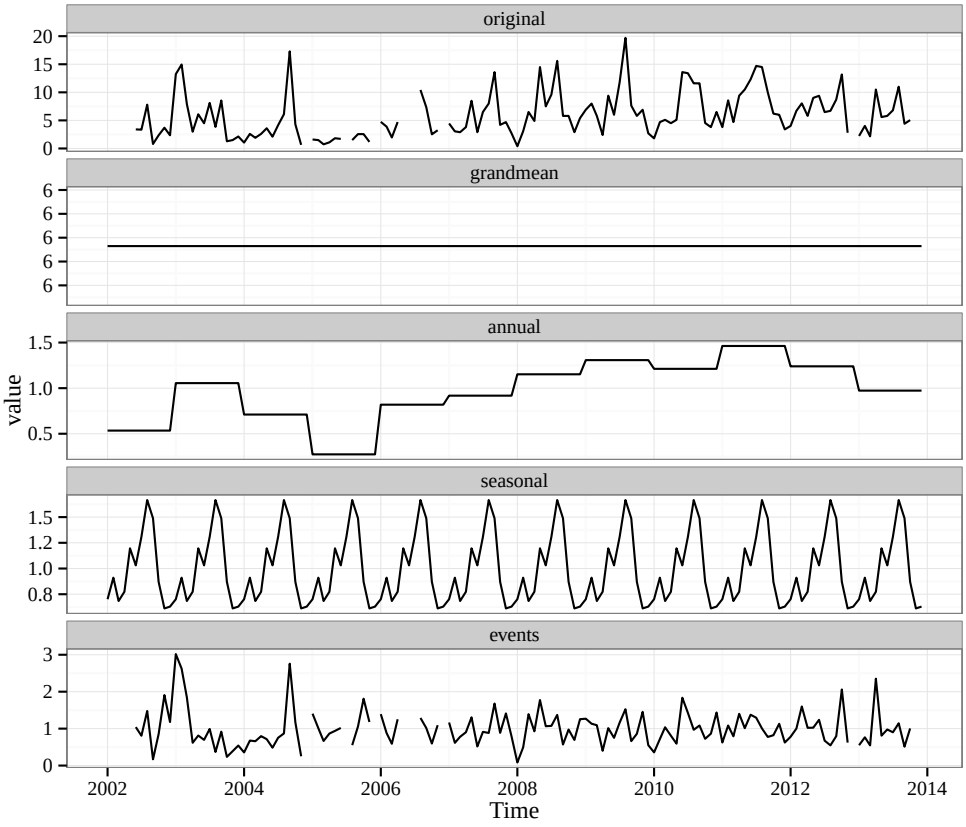

Fig. 4. Additive decomposition of a multi-year chlorophyll time series into the grandmean, annual, seasonal, and events components using the `decomp_cj` function.

Several graphics showing seasonal and annual trends for a given SWMP parameter can be obtained using the `plot_summary` function. The 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 for the parameter. 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, whereas weather and water quality data are based on daily averages. Cumulative precipitation data are based on the daily maximum. The function returns a graphics object (Grob) of multiple ggplot objects. An interactive Shiny application [18] that uses this function is available (see the Supporting Information).

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

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

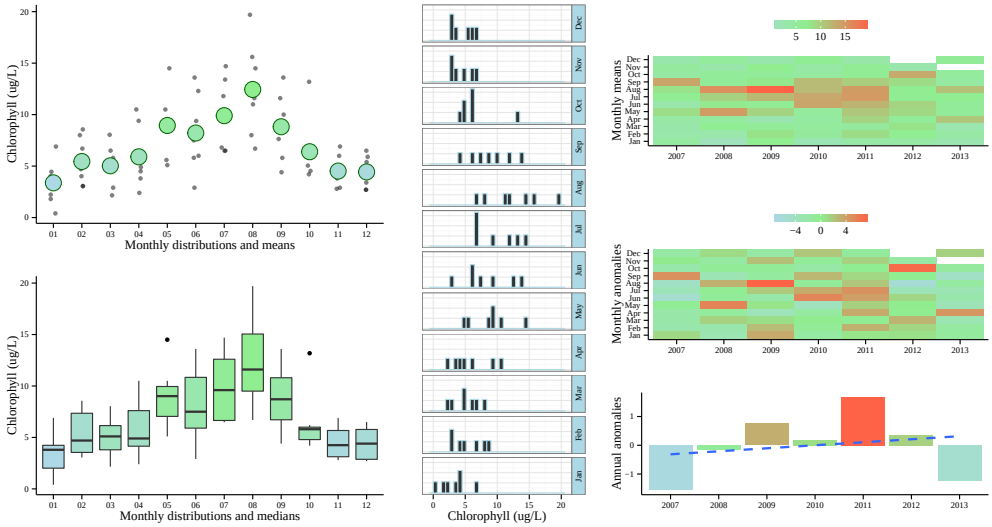

Fig. 5. Summaries of a multi-year chlorophyll time series using the `plot_summary` function. Summaries include monthly distributions (means on top left, quantiles on bottom left), monthly histograms (center), monthly means by year (top right), deviation from monthly means (middle right), and annual trends as deviations from the grand mean (bottom right)

Estimates of ecosystem metabolism provide a useful measure of overall system productivity. These estimates are commonly used to evaluate whether an ecosystem is a net source or sink of organic material. The open-water method [19] 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 [20], Thebault et al. [21]). The diffusion-corrected dissolved oxygen (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

The `ecometab` function is used to implement an adaptation of the open-water method [19, 22]. Several assumptions must be met for a valid interpretation of the results. In general, the dissolved oxygen time series is assumed to represent the same water mass over time. Tidal advection may have a significant influence on the time series, which can contribute to a significant amount of noise in metabolic estimates. The

extent to which tidal advection influences the dissolved oxygen signal depends on various site-level characteristics and an intimate knowledge of the site may be required. 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 water quality 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. Additionally, 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. Other assumptions may apply and the user should consult the relevant literature. All calculations within the function are done using molar units (e.g., mmol O^2 m^{-3}). The output can be returned as mass units by changing the default argument. Input data must be in standard mass units for DO (mg L-1)

The following is an example that shows use of the function from a combined water quality and weather data set. The results can be plotted using `plot_metab` (Fig. 6).

```
## 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, trace = FALSE)
plot_metab(res)
```

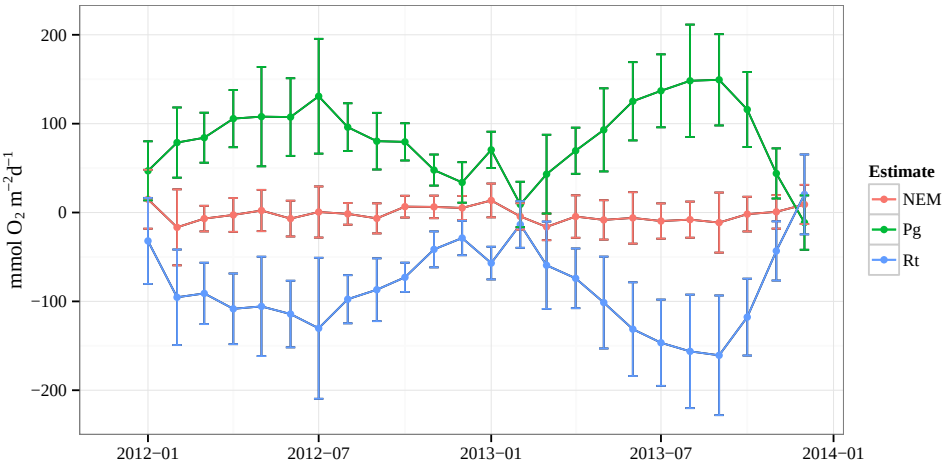

Fig. 6. Monthly aggregations of ecosystem metabolism estimates (net ecosystem metabolism, gross production, and total respiration) for combined water quality and weather data at Apalachicola Bay, Florida.

Table 4. Miscellaneous functions available from the SWMPr package. Most are used within the main functions above but may be useful for more customized evaluationg of SWMP data. Full documentation for each function is in the help file (e.g., execute `?calckl` at the command line).

Function	Description
`calckl`	Estimate the reaeration coefficient for air-sea gas exchange. This is only used within the `ecometab` function.
`map_reserve`	Create a map of all stations in a reserve using the ggmap package.
`metab_day`	Identify the metabolic day for each approximate 24 period in an hourly time series. This is only used within the `ecometab` function.
`param_names`	Returns column names as a list for the parameter type(s) (nutrients, weather, or water quality). Includes QAQC columns with `f_` prefix. Used internally in other functions.
`parser`	Parses html returned from CDMO web services, used internally in retrieval functions.
`swmpr`	Creates object of `swmpr` class, used internally in retrieval functions.
`time_vec`	Converts time vectors to `POSIXct` objects with correct time zone for a site/station, used internally in retrieval functions.

Miscellaneous functions

Several additional functions are provided that do not fit the above categories (Table 4). These functions are generally used within the main functions but may be useful for more customized evaluation of SWMP data.

For brevity, only the `reserve_map` function is discussed. This function can be used to create a map with all stations at a reserve by passing arguments to functions in the ggmap package [23]. The current function is limited to Google maps of four types that can be set with the `map_type` argument: terrain (default), satellite, roadmap, or hybrid. The `zoom` argument may have to be chosen through trial and error depending on the spatial extent of the reserve. See the help documentation for the `ggmap` function for more info on `zoom`.

```
# plot the stations at Jacques Cousteau reserve
map_reserve('jac')
```

Applications using the SWMPr package

The ability to evaluate environmental characteristics between estuaries within the NERRS program has been greatly improved using functions in the SWMPr package. This section describes two examples of applications using the SWMPr package to illustrate the improved ability to synthesize and evaluate multi-year time series of estuarine data. First, the open-water method for estimating metabolism was applied to nearly all co-located water quality and weather sites at the NERRS reserve for all years of available data. The results are provided primarily to illustrate ease of use of the functions and secondarily to provide an update to results described in Caffrey [10] and Caffrey [11]. A comprehensive evaluation of metabolic rates between estuaries has not

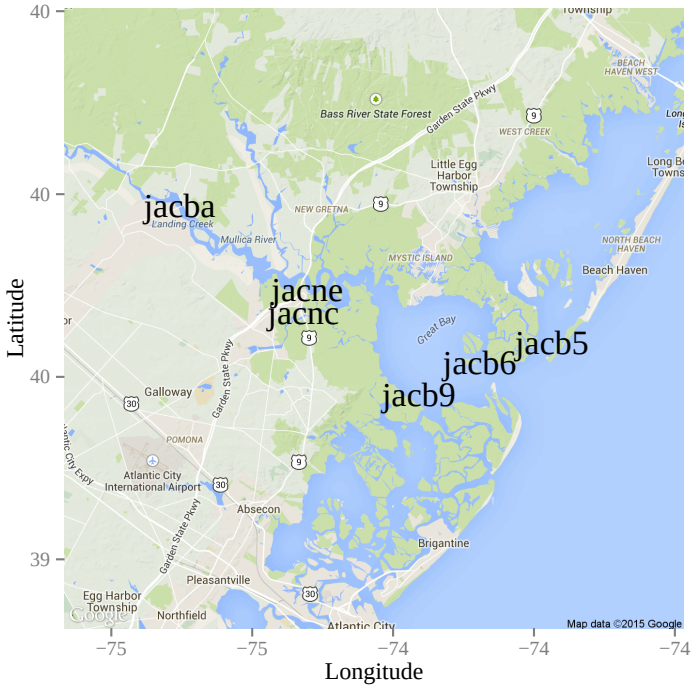

Fig. 7. Locations of all sites at the Jacques Cousteau reserve using the `map_reserve` function.

been conducted using the most recent SWMP data. The results are presented to facilitate additional research to evaluate factors that influence variation between sites.

Caffrey [10] and Caffrey [11] describe the theory and application of the open-water method to estimate ecosystem metabolism using two sites at each of the NERRS reserve. Time series data included approximately five years of half-hour water quality observations at each site. Additionally, the air-sea gas exchange model, as implemented by the current `ecometab` function (see Ro and Hunt [20] and Thebault et al. [21]), was not incorporated into the initial metabolism estimates such that a constant value for the reaeration coefficient was assumed. This coefficient provides an estimate of the rate of air-sea gas exchange that varies as a function of wind speed, temperature, barometric pressure, salinity, and depth of the water column The inclusion of weather data in the calculation allows for a more precise estimate of air-sea gas exchange and consequently more reliable estimates of ecosystem metabolism (see Caffrey et al. [22] for details).

All water quality and weather observations for all NERRS sites were obtained through a bulk data request in November of 2014 using the zip downloads feature of CDMO. After the download was complete, all csv files for each station were imported into R using the `import_local` function and then saved again on a local hard drive as a binary .RData file. This resulted in a single `swmpr` object for each site. All files were then uploaded to a remote server. An R script was executed that retrieved and processed the `swmpr` objects for each site using the following functions in sequence:

Summary

Supporting Information

Trends in SWMP parameters

https://beckmw.shinyapps.io/swmp_comp This widget is an interactive tool to evaluate trends in SWMP data within and between sites. Trends are described by an increase or decrease in values over time using a simple linear regression of summarized data. The regression for each station can be viewed by clicking on a map location. Trends at each station are plotted as circles that identify the direction and significance of the trend. The trend direction is blue for decreasing and red for increasing. The significance is indicated by radius of the circle and color shading where larger points with darker colors indicate a strong trend.

Monthly and annual summary of SWMP parameters

https://beckmw.shinyapps.io/swmp_summary/

This interactive widget provides graphical summaries of water quality, weather, and nutrient station data from SWMP. The drop down menus can be used to select the station, date range, and parameter for plotting. The raw data used for plotting include all SWMP records from the earliest date at each station after processing to remove QAQC flags. The data were downloaded from the CDMO web services on November 25th, 2014 and include observations up to that date. Plots are based on daily averages for each parameter. Cumulative precipitation data are based on the daily maximum.

Acknowledgments

I acknowledge the significant efforts of NERRS researchers and staff for providing access to high-quality monitoring data. Thanks particularly to Dwayne Porter and Melissa Ide from CDMO for maintaining the online database. Thanks to Marie Bundy and Nikki Dix for providing me the opportunity to share this package with the broader NERRS community. Thanks to Todd O'Brien for the inspiration for the online widgets in the supporting information. Thanks to Mike Murrell and Jim Hagy III for assistance with documentation and implementation of the ecosystem metabolism functions. The views expressed in this article are those of the authors and do not necessarily reflect the views or policies of the U.S. Environmental Protection Agency.

References

1. Glasgow HB, Burkholder JM, Reed RE, Lewitus AJ, Kleinman JE. Real-time remote monitoring of water quality: a review of current applications, and advancements in sensor, telemetry, and computing technologies. Journal of Experimental Marine Biology and Ecology. 2004;300(1-2):409–448.

2. Fries DP, Ivanov SZ, Bhanushali PH, wilson JA, Broadbent HA, Sanderson AC. Broadband, low-cost, coastal sensor nets. Oceanography. 2008;20(4):150–155.

3. Reed RE, Burkholder JM, Allen EH. Current online monitoring technology for surveillance of algal blooms, potential toxicity, and physicalechemical structure in rivers, reservoirs, and lakes. In: American Water Works Association Manual M57, Algae. Denver, Colorado: American Water Works Association; 2010. p. 1–24.

4. National Weather Service, National Oceanic and Atmospheric Administration. Hydrometeorological Automated Data System website; 2015. http://www.nws.noaa.gov/oh/hads/. (Accessed March, 2015).

5. Sanders CA, Rodriguez M, Greenbaum E. Stand-off tissue-based biosensors for the detection of chemical warfare agents using photosynthetic fluorescence induction. Biosensors and Bioelectronics. 2008;16(7-8):439–446.

6. Campbell JL, Rustad LE, Porter JH, Taylor JR, Dereszynski EW, Shanley JB, et al. Quantity is nothing without quality: Automated QA/QC for streaming environmental sensor data. BioScience. 2013;63(7):574–585.

7. Millie DF, Weckman GR, Young WA, Ivey JE, Fries DP, Ardjmand E, et al. Coastal 'big data' and nature-inspired computation: prediction potentials, uncertainties, and knowledge derivation of neural netowrks for an algal metric. Estuarine, Coastal and Shelf Science. 2013;125:57–67.

8. Bulthius DA. Distribution of seagrasses in a north Puget Sound estuary - Padilla Bay, Washington, USA. Aquatic Botany. 1995;50(1):99–105.

9. Dix NG, Phlips EJ, Gleeson RA. Water quality changes in the Guana Tolomato Matanzas National Estuarine Research Reserve, Florida, associated with four tropical storms. Journal of Coastal Research. 2008;55(SI):26–37.

10. Caffrey JM. Production, respiration and net ecosystem metabolism in U.S. estuaries. Environmental Monitoring and Assessment. 2003;81(1-3):207–219.

11. Caffrey JM. Factors controlling net ecosystem metabolism in U.S. estuaries. Estuaries. 2004;27(1):90–101.

12. Sanger DM, Arendt MD, Chen Y, Wenner EL, Holland AF, Edwards D, et al. A synthesis of water quality data: National Estuarine Research Reserve System-wide Monitoring Program (1995-2000). Charleston, South Carolina: National Estuarine Research Reserve Technical Report Series 2002:3. South Carolina Department of Natural Resources, Marine Resources Division Contribution No. 500; 2002.

13. Wenner E, Sanger D, Arendt M, Holland AF, Chen Y. Variability in dissolved oxygen and other water-quality variables within the National Estuarine Research Reserve System. Journal of Coastal Research. 2004;45(SI):17–38.

14. System-Wide Monitoring Program Data Analysis Training. SWMP Data Analysis Training Workshop provided at the 2014 NERRS/NERRA Annual Meeting, November 17, 2014; 2014. http://copepod.org/nerrs-swmp-workshop/.

15. RDCT (R Development Core Team). R: A language and environment for statistical computing, v3.1.2. R Foundation for Statistical Computing, Vienna, Austria; 2014. http://www.R-project.org.

16. Wickham H. Advanced R. Boca Raton, Florida: Chapman and Hall, CRC Press; 2014.

17. Cloern JE, Jassby AD. Patterns and scales of phytoplankton variability in estuarine-coastal ecosystems. Estuaries and Coasts. 2010;33(2):230–241.

18. Chang W, Cheng J, Allaire J, Xie Y, McPherson J. shiny: Web Application Framework for R; 2015. R package version 0.11.1. Available from: http://CRAN.R-project.org/package=shiny.

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

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

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

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

23. Kahle D, Wickham H. ggmap: A package for spatial visualization with Google Maps and OpenStreetMap; 2013. R package version 2.3. Available from: http://CRAN.R-project.org/package=ggmap.