SWMPr: An R Package for Retrieving, Organizing, and Analyzing Environmental Data for Estuaries

by Marcus W Beck

Abstract The System-Wide Monitoring Program (SWMP) was implemented in 1995 by the US National Estuarine Research Reserve System. This program has provided two decades of continuous monitoring data at over 140 fixed stations in 28 estuaries. Simple tools for processing and evaluating the increasing quantity of data provided by the monitoring network have prevented broad-scale comparisons between systems and, in some cases, simple trend analysis of water quality parameters at individual sites. This article describes the **SWMPr** package that provides several functions to facilitate data retrieval, organization, and analysis of time series data in the reserve estuaries. Previously unavailable functions for estuaries are also provided to estimate rates of ecosystem metabolism using the open-water method. The **SWMPr** package has facilitated a cross-reserve comparison of water quality trends and provides an effective approach to link quantitative information with analysis tools that will inform coastal protection and restoration.

Introduction

The development of low-cost, automated sensors that collect data in near real-time has enabled a proliferation of standardized environmental monitoring programs (Glasgow et al., 2004; Fries et al., 2008). An invaluable source of monitoring data for coastal regions in the United States is provided by the National Estuarine Research Reserve System (NERRS, `http://www.nerrs.noaa.gov/`). This network of 28 estuary reserves was created to address long-term research, monitoring, education, and stewardship goals in support of coastal management. The System-Wide Monitoring Program (SWMP) was implemented in 1995 at over 140 stations across the reserves to provide a robust, long-term monitoring system for water quality, weather, and land-use/habitat change. The SWMP network provides a continuous source of data collected at near real-time with the potential to evaluate causes of spatiotemporal variation in environmental condition and ecosystem function. Environmental researchers have expressed a need for quantitative analysis tools to evaluate trends in water quality time series given the quantity and quality of data provided by SWMP (System-Wide Monitoring Program Data Analysis Training, 2014).

This article describes the **SWMPr** package that was developed for estuary monitoring data from the System-Wide Monitoring Program. Functions provided by **SWMPr** address many 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, web applications derived from **SWMPr** and **shiny** illustrate potential applications using the functions in this package. The software is provided specifically for use with NERRS data, although many of the applications are relevant for addressing common challenges working with large environmental datasets.

SWMP overview and data retrieval

The **SWMPr** package is developed for the continuous abiotic monitoring network that represents a majority of SWMP data and, consequently, the most challenging to evaluate. 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 accessible through the Centralized Data Management Office (CDMO) web portal (`http://cdmo.baruch.sc.edu/`), where multiple quality assurance/quality control (QAQC) measures are used to screen the information for accuracy and reliability. The final data include all timestamped observations including relevant QAQC flags with the appropriate qualifier.

At the time of writing, the CDMO web portal provides over 60 million water quality, weather, and

Table 1: Retrieval functions available from the **SWMPr** package. Full documentation for each function is in the help file (e.g., execute `?all_params` for individual functions or `help.search('retrieve', package = 'SWMPr')` for all).

Function	Description
`all_params`	Retrieve 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. Wrapper to `exportAllParamsDateRangeXMLNew`.
`import_local`	Import files from a local path. The files must be in a specific format, such as those returned from the CDMO using the zip downloads option.
`single_param`	Retrieve 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.

nutrient records that have been authenticated through systematic QAQC procedures. Records for each station 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 ('wq') for the East Bay station ('eb') at the Apalachicola reserve ('apa'). Similarly, a suffix of 'met' or 'nut' specifies the weather (meteorological) or nutrient stations. All reserve names, stations, and date ranges for each parameter type can be viewed on the CDMO website. Alternatively, the `site_codes` (all sites) or `site_codes_ind` (single site) functions provided by **SWMPr** can be used. As noted below, an IP address must be registered with CDMO before using the data retrieval functions in **SWMPr**. Web services are provided by CDMO for 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.

Structure of the SWMPr package

SWMPr functions are categorized by one of three steps in the data workflow: ***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 for working with a `"swmpr"` object. The following describes the package structure, beginning with the retrieval functions, a description of the `"swmpr"` object returned after retrieval, and, finally, the organizing and analyzing functions.

Data retrieval

SWMPr can import data into R through direct download from the CDMO or by importing local data that was previously downloaded (Table 1). The IP address for the computer making the request must be registered if the first approach is used (see CDMO website). 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')
```

Retrieval functions to import data directly into R from the CDMO include `all_params`, `all_params_dtrng`, and `single_param`. Due to rate limitations on the CDMO server, the retrieval functions return a limited number of records with each request. However, the **SWMPr** functions use the native CDMO web

services iteratively (i.e., within a loop) to obtain all requested records. Download time can be excessive for longer time series.

```
> # 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')
```

The second approach for data retrieval is to use the `import_local` function to import data into R after downloading from CDMO. This approach is most appropriate for large, specific data requests. The `import_local` function is designed for data from the zip downloads feature in the advanced query section of the CDMO website. The zip downloads feature can be used 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 to import files directly from the zipped folder or after the folder is decompressed.

Occasionally, non-unique observations are present in the raw data. These duplicates may be actual replicates with unique time stamps, such as replicate samples for monthly nutrient data, or erroneous duplicates with non-unique time stamps. The `import_local` function handles duplicate entries differently depending on the data type. For water quality and nutrient data, duplicate time stamps are simply removed. Nutrient data often contain replicate samples with similar but not identical time stamps within the span of a few minutes. Nutrient data with replicates with unique time stamps are not removed but can be further processed using `rem_reps`. Weather data prior to 2007 may also contain duplicate time stamps at frequencies for hourly (denoted as '60') and daily ('144') averages, in addition to 15 minute frequencies. Only duplicate values at 15 minutes are averaged for weather data during import.

```
> # import local data for apaebmet
>
> # this is an example path with the decompressed csv files
> path <- 'C:/my_path'
>
> # this is an example path for zipped csv files
> path <- 'C:/my_path.zip'
>
> # import, do not include file extension
> import_local(path, 'apadbwq')
```

The `"swmpr"` object class

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 and additional parameters for a given data type (weather, nutrients, or water quality). Corresponding QAQC columns for each parameter are also returned if provided by the initial data request. The following shows an example of the raw data imported using `all_params`.

```
> # import all paramaters for the station
> # three most recent records
> exdat <- all_params('apadbwq', Max = 3, trace = F)
> exdat
##         datetimestamp temp f_temp spcond f_spcond sal f_sal do_pct
## 1 2015-08-18 07:00:00   28      0     34        0  21     0     77
## 2 2015-08-18 07:15:00   28      0     34        0  21     0     75
## 3 2015-08-18 07:30:00   28      0     33        0  21     0     82
```

Table 2: Attributes of a "swmpr" object that describe characteristics of the data.

Attributes	Class	Description
names	character	Column names of the entire data set, inherited from the data.frame object class.
row.names	integer	Row names of the data set, inherited from the data.frame object class.
class	character	Class of the data object indicating "swmpr" and data.frame.
station	character	Station identifier used by NERRS as a string with 7 or 8 characters.
parameters	character	Character vector of column names for data parameters, e.g., `do_mgl', `turb', etc.
qaqc_cols	logical	Indicates if QAQC columns are present in the raw data.
date_rng	POSIXct	Start and end dates for the data.
timezone	character	Timezone of the station using the city/country format[a].
stamp_class	character	Class of the datetimestamp column, usually POSIXct unless data have been aggregated.

[a]Time zones that do not observe daylight savings are used for "swmpr" objects and may not be cities in the United States. For example, "America/Jamaica" is used for Eastern Standard Time.

```
##   f_do_pct do_mgl f_do_mgl depth f_depth ph f_ph turb f_turb chlfluor
## 1        0      5        0     2       0  8    0   13      0       NA
## 2        0      5        0     2       0  8    0    9      0       NA
## 3        0      6        0     2       0  8    0   11      0       NA
##   f_chlfluor level f_level cdepth clevel f_cdepth f_clevel
## 1         -2    NA      -1     NA     NA       -1
## 2         -2    NA      -1     NA     NA       -1
## 3         -2    NA      -1     NA     NA       -1
```

The attributes for a "swmpr" object are descriptors that are appended to the raw data (Table 2). These act as metadata that are used internally by many of the package functions and are updated as the data are processed. The attributes are not visible with the raw data but can be viewed as follows.

```
> # import sample data from package
> data(apadbwq)
> dat <- apadbwq
>
> # view all attributes of dat
> attributes(dat)
>
> # view a single attribute of dat
> attr(dat, 'station')
```

The "swmpr" object class was created for use with the organizing and analyzing functions. This object-oriented approach is standard for R (i.e., the S3 object system), such that specific methods for generic functions are developed for the object class. A "swmpr" object also secondarily inherits methods from the data.frame class, such that common data.frame methods available in R also apply to "swmpr" objects. Available methods for the "swmpr" class are described below and can also be viewed:

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

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. These data are also included with the package as binary data files (RData) that can be loaded using the data function, as

Table 3: Organizing functions available from the **SWMPr** package. Full documentation for each function is in the help file (e.g., execute `?comb` for individual functions or `help.search('organize', package = 'SWMPr')` for all).

Function	Description
`comb`	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`	Remove QAQC columns and remove data based on QAQC flag values for a `"swmpr"` object. Only applies if QAQC columns are present.
`qaqcchk`	View a summary of the number of observations in a `"swmpr"` object that are assigned to each QAQC flag used by CDMO. The output can be used to inform further processing.
`rem_reps`	Remove replicate nutrient data that occur on the same day. The default is to average replicates.
`setstep`	Format data from a `"swmpr"` object to a continuous time series at a given timestep. The function is also used in `comb`.
`subset`	Subset by dates and/or columns for a `"swmpr"` object. This is a method passed to the generic `subset` function provided in the base installation.

shown above. These include `"swmpr"` objects for four stations at Apalachicola Bay: apacpnut, apacpwq, apadbwq, and apaebmet. Information for each file can be viewed in the help documentation (e.g., `?apacpnut`).

Data organizing

The organize functions are used to clean or prepare the imported data for analysis, including viewing and removal of QAQC flags, subsetting, combining replicate nutrient observations, creating a standardized time series, and combining data of different types (Table 3).

The `qaqc` function is a simple screen to retain observations from the data with specified QAQC flags (see http://cdmo.baruch.sc.edu/data/qaqc.cfm). Each parameter in the imported `"swmpr"` object will have a corresponding QAQC column of the same name with the added prefix `f_` (e.g., `do_mgl`, `f_do_mgl`). Values in the QAQC column range from -5 to 5 to indicate the QAQC flag that was assigned by CDMO during initial processing. The `qaqc` function is used to remove observations in the raw data with given flags, with the default option to retain only values with the `0` QAQC flag (i.e., passed initial CDMO checks). 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. The function returns the original data with the QAQC columns removed and `NA` (not available) values for observations that do not meet the criteria specified in the function call.

```
> # 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.

```
> # 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` function combines nutrient data that occur on the same day to preserve an approximate monthly 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 defaults 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, filter qaqc flags
> data(apacpnut)
> dat <- apacpnut
> dat <- qaqc(dat)
>
> # remove replicate nutrient data
> rem_reps(dat)
>
> # use different function to aggregate replicates
> func <- function(x) max(x, na.rm = T)
> rem_reps(dat, FUN = func)
```

A subset method added to the existing generic `subset` function in R 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 as a character string passed to the `operator` 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 (e.g., '2007-01-01 06:30'). 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.

```
> # import data
> data(apaebmet)
> dat <- apaebmet
>
> # select two parameters from dat
> subset(dat, select = c('rh', 'bp'))
>
> # subset records greater than or equal to a date
> subset(dat, subset = '2013-01-01 0:00', operator = '>=')
>
> # subset records within a date range
> subset(dat, subset = c('2012-07-01 6:00', '2012-08-01 18:15'))
>
> # subset records within a date range, select two parameters
> subset(dat, subset = c('2012-07-01 6:00', '2012-08-01 18:15'),
+   select = c('atemp', 'totsorad'))
>
> # remove rows/columns that do not contain data
> subset(dat, rem_rows = T, rem_cols = T)
```

The `setstep` function formats a `"swmpr"` object to a continuous time series at a given time step. This function is usually not necessary because most stations collect data at 15 minute intervals, but it can be used to combine data or convert an existing time series to a different 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 the 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. Time steps that do not match any existing data within the limits of the `differ` argument are not discarded, although a corresponding data value will not be assigned.

```
> # import, qaqc removal
> data(apadbwq)
> dat <- qaqc(apadbwq)
>
> # 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 setstep function is used internally such that timestep and differ are accepted arguments for comb. All arguments must be called explicitly since an arbitrary number of "swmpr" objects can be used as input to comb. 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 from the earliest and latest dates for all input objects. Passing `intersect' to method will create a continuous time series 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 data based on a continuous time series for the specified station, which must be present in the input data. Currently, combining identical data types from different stations is not possible (e.g., two water quality stations from the same reserve).

```
> # get nut, wq, and met data as separate objects
> data(apacpnut)
> data(apacpwq)
> data(apaebmet)
> swmp1 <- apacpnut
> swmp2 <- apacpwq
> swmp3 <- apaebmet
>
> # combine nut and wq data by union
> comb(swmp1, swmp2, method = 'union')
>
> # combine nut and wq data by intersect
> comb(swmp1, swmp3, method = 'intersect')
>
> # combine nut, wq, and met data by nut 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 4). The general purpose tools are "swmpr" methods that were developed for existing generic functions in the R base installation or relevant packages (**SWMPr** imports and dependencies are listed on CRAN). These functions include "swmpr" methods for aggreswmp, filter, and approx to deal with missing or noisy data and more general functions for exploratory data analysis, such as plot, lines, and hist methods. Decomposition functions, decomp and decomp_cj, are provided as relatively simple approaches for decomposing time series into additive or multiplicative components. Functions to estimate and plot ecosystem metabolism from combined water quality and weather data are provided by the ecometab and plot_metab functions. 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 aggreswmp function aggregates parameter data for a "swmpr" object by set units of time. 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 by a user-defined function, which defaults to the mean. A "swmpr" object is returned for the aggregated

Table 4: Analysis functions available from the **SWMPr** package. Full documentation for each function is in the help file (e.g., execute `?aggreswmp` for individual functions or `help.search('analyze', package = 'SWMPr')` for all).

Function	Description
`aggreswmp`	Aggregate `"swmpr"` objects for different time periods - years, quarters, months, weeks, days, or hours. The aggregation function defaults to the mean.
`aggremetab`	Aggregate metabolism data from a `"swmpr"` object. This is primarly used within `plot_metab` but may be useful for simple summaries of daily metabolism data.
`ecometab`	Estimate ecosystem metabolism for a combined water quality and weather dataset using the open-water method.
`decomp`	Decompose a `"swmpr"` time series into trend, seasonal, and residual components. This is a simple wrapper to `decompose` (Kendall and Stuart, 1983). Decomposition of monthly or daily trends is possible.
`decomp_cj`	Decompose a `"swmpr"` time series into grandmean, annual, seasonal, and events components. This is a simple wrapper to `decompTs` in the **wq** package (Jassby and Cloern, 2014). Only monthly decomposition is possible.
`hist`	Plot a histogram for a `"swmpr"` object.
`lines`	Add lines to an existing plot created with `plot`.
`map_reserve`	Create a map of all stations in a reserve using the **ggmap** package (Kahle and Wickham, 2013).
`na.approx`	Linearly interpolate missing data (`NA` values) in a `"swmpr"` object. The maximum gap size that is interpolated is defined by the arguments.
`overplot`	Plot multiple time series on the same y-axis.
`plot`	Plot a univariate time series for a `"swmpr"` object. The parameter name must be specified.
`plot_metab`	Plot ecosystem metabolism estimates after running `ecometab` on a combined `"swmpr"` object.
`plot_summary`	Create summary plots of seasonal/annual trends and anomalies for a water a single paramter of interest.
`smoother`	Smooth `"swmpr"` objects with a moving window average. Window size and sides (e.g., centered) can be specified, passed to `filter`.

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 created in the **data.table** package (Dowle et al., 2014). 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, particularly for nutrient data (see the example below).

```
> # get data, keep all observations
> data(apacpnut)
> dat <- qaqc(apacpnut, qaqc_keep = NULL)
>
> # aggregate by quarters
> agg_dat <- aggreswmp(dat, by = 'quarters')
> nrow(agg_dat)
## [1] 47
> # aggregate by quarters, remove rows with NA values
> # note the reduction in the number of rows
> agg_dat2 <- aggreswmp(dat, by = 'quarters', na.action = na.omit)
> nrow(agg_dat2)
```

```
## [1] 16
```

Time series can be smoothed to better characterize a signal from noisy data (Figure 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. The `window` argument specifies the number of observations included in the moving average where larger windows result in greater smoothing. The `sides` argument specifies how the average is calculated for each observation. Setting `sides = 1` will filter observations within the window that are previous to the current observation, whereas `sides = 2` will filter observations within the window centered at zero lag from the current observation. As before, the `params` argument specifies which parameters to smooth.

```
> # import data, qaqc and subset
> data(apadbwq)
> dat <- qaqc(apadbwq)
> dat <- subset(dat, select = 'do_mgl',
+   subset = c('2012-07-09 00:00', '2012-07-24 00:00')
+   )
>
> # smooth using a window of 50 observations
> dat_smooth <- smoother(dat, window = 50, params = 'do_mgl')
>
> # plot raw and smoothed
> plot(dat)
> lines(dat_smooth, col = 'red', lwd = 2)
```

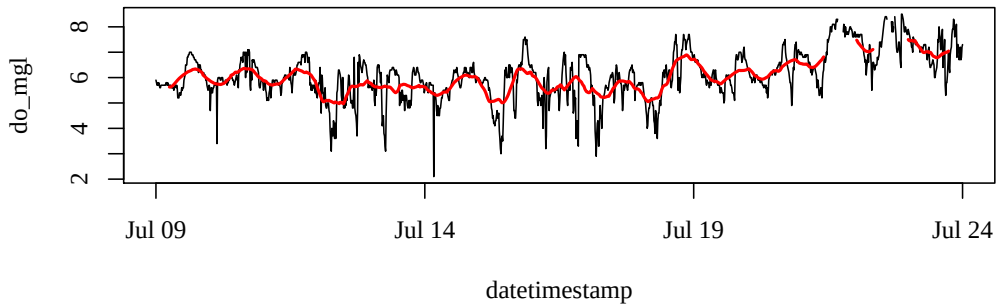

Figure 1: Raw and smoothed dissolved oxygen data for a two-week period after using the `smoother` function.

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. An approach implemented in **SWMPr** is linear interpolation using the `na.approx` function (Figure 2). A simple curve fitting method is used to create a continuous set of records between observations separated by missing data. However, the ability of the interpolated data to approximate actual trends is related to the maximum gap size between observations with missing data. Interpolation between larger gaps are less likely to resemble patterns of an actual parameter, whereas interpolation between smaller gaps are often more accurate. An upper limit on the maximum gap size to interpolate trends depends on the characteristics of the dataset such that a trial and error approach is appropriate for most applications. The `maxgap` argument passed to `na.approx` defines the maximum gap size for interpolation and the following illustrates use of different maximum values to fill missing data.

```
> # get data, qaqc and subset
> 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(dat, main = 'Raw')
> 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)
```

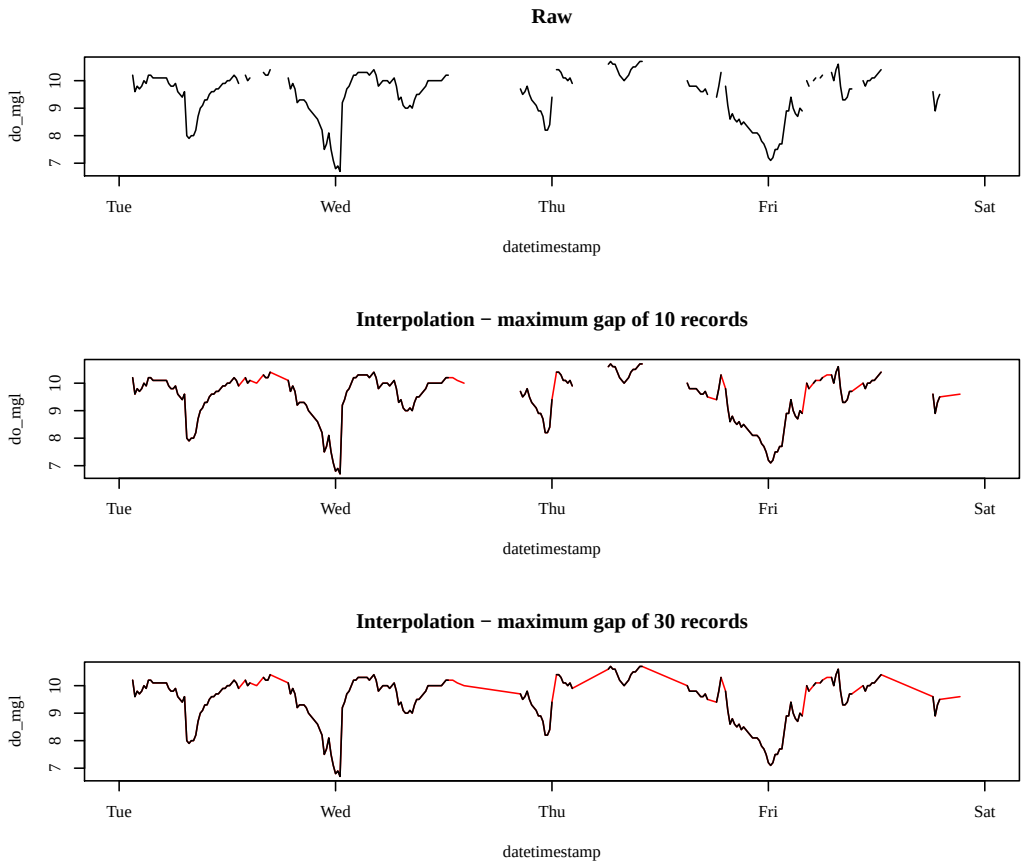

Figure 2: Examples illustrating use of the `na.approx` function to fill gaps of different sizes in a dissolved oxygen time series for a four day period.

The disaggregation of time series into additive or multiplicative components that can be attributed to separate sources of variance is another common application for trend analysis. The `decomp` function is a simple wrapper to `decompose` (Kendall and Stuart, 1983) that separates a time series into components describing a trend, cyclical variation (e.g., daily or annual), and the remainder (Figure 3). An additive decomposition assumes that the cyclical component of the time series is stationary (i.e., the variance is constant), otherwise a multiplicative decomposition can be used. The `frequency` argument describes the periodicity of the cyclical parameter in units of the native time step. For example, the `frequency` for a parameter with daily 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, character strings of `'daily'` or `'annual'` can be supplied in place of numeric values, although any number can be used to identify an arbitrary cyclical component. A starting value of the time series must be supplied in the

latter case that indicates the sequence in the cycle for the first observation. For example, the starting value would be 1 if the first observation is at sunrise for a diurnal cycle (see the help file for the `ts` function for details). Use of the `setstep` function is also required to standardize the time step prior to decomposition.

```
> # get data
> data(apadbwq)
> dat <- apadbwq
>
> # subset for daily decomposition
> dat <- subset(dat, subset = c('2013-07-01 00:00', '2013-07-31 00:00'))
>
> # daily decomposition of DO and plot
> dc_dat <- decomp(dat, param = 'do_mgl', frequency = 'daily')
> plot(dc_dat)
```

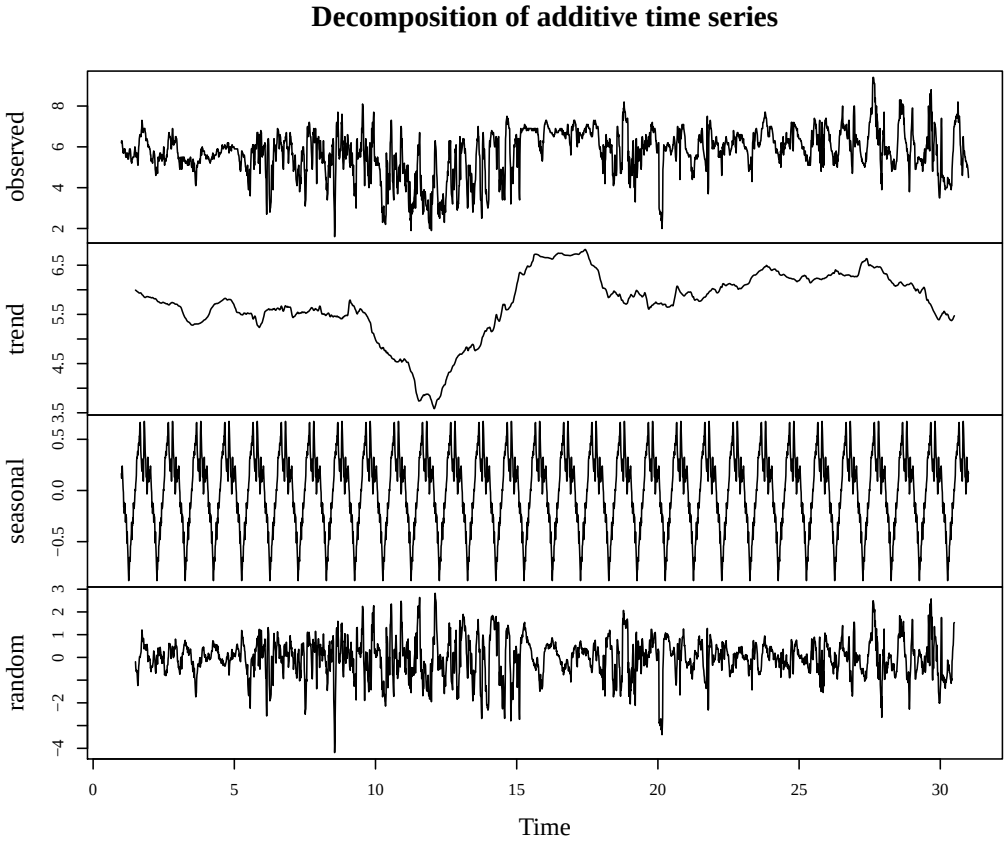

Figure 3: An additive decomposition of dissolved oxygen into a trend, seasonal (daily), and random component using the `decomp` function.

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 (Cloern and Jassby, 2010; Jassby and Cloern, 2014). 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 be aggregated on the monthly scale before decomposition. The time series is decomposed into the grandmean, annual, seasonal, and events components, as compared to trend, seasonal, and random components for the `decomp` function described above. For both, the random or events components can be considered anomalies that do not follow the trends in the remaining categories. 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. Figure 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
> decomp_cj(dat, param = 'chla_n')
```

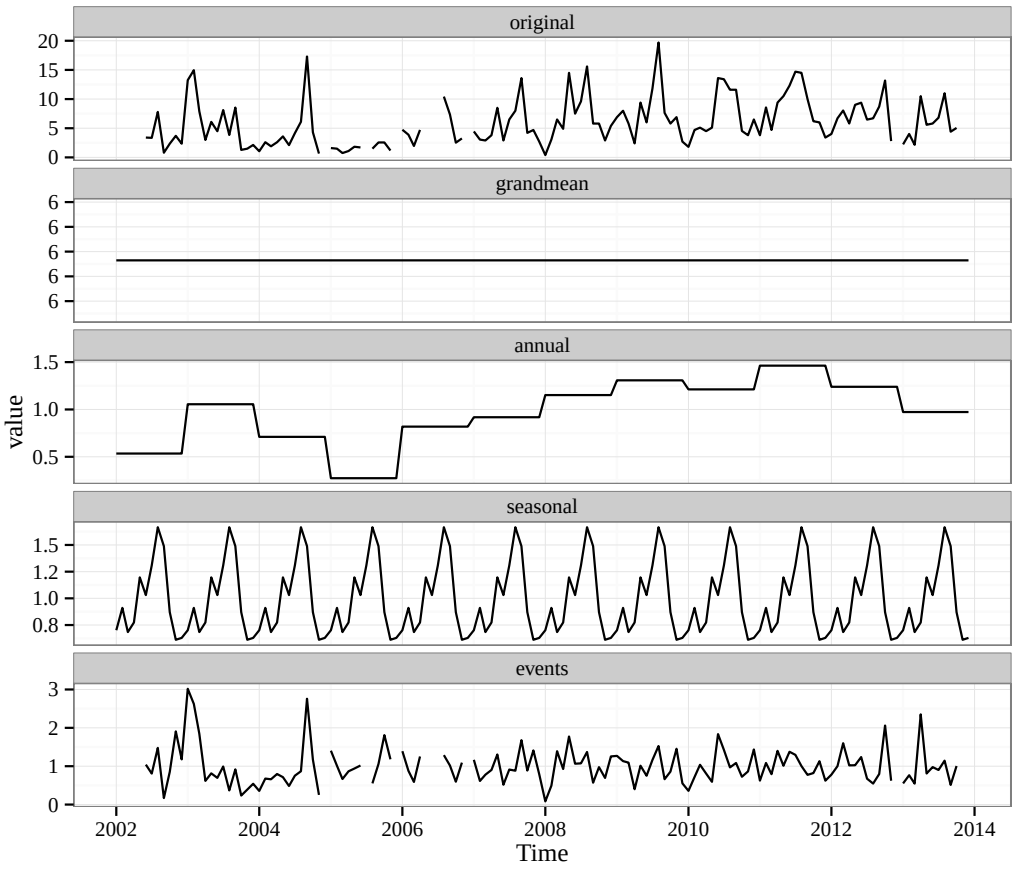

Figure 4: Additive decomposition of a multi-year chlorophyll time series into the grandmean, annual, seasonal, and events components using the `decomp_cj` function.

Detailed exploratory graphics are also useful for evaluating general trends in observed data. Several graphics showing seasonal and annual trends for a single SWMP parameter can be obtained using the `plot_summary` function (Figure 5). The plots include monthly distributions, monthly anomalies, and annual anomalies in multiple formats. An interactive Shiny web application (Chang et al., 2015) that uses this function is available for viewing results of any parameter at all SWMP sites (see the Applications using the **SWMPr** package section).

```
> ## import data
> data(apacpnut)
> dat <- qaqc(apacpnut)
>
> ## plot
> plot_summary(dat, param = 'chla_n', years = c(2007, 2013))
```

The `overplot` function provides an alternative approach to viewing observed data from the same station. This function uses the base **graphics** package to plot multiple time series on the same y-axis.

```
> ## import data
> data(apacpwq)
```

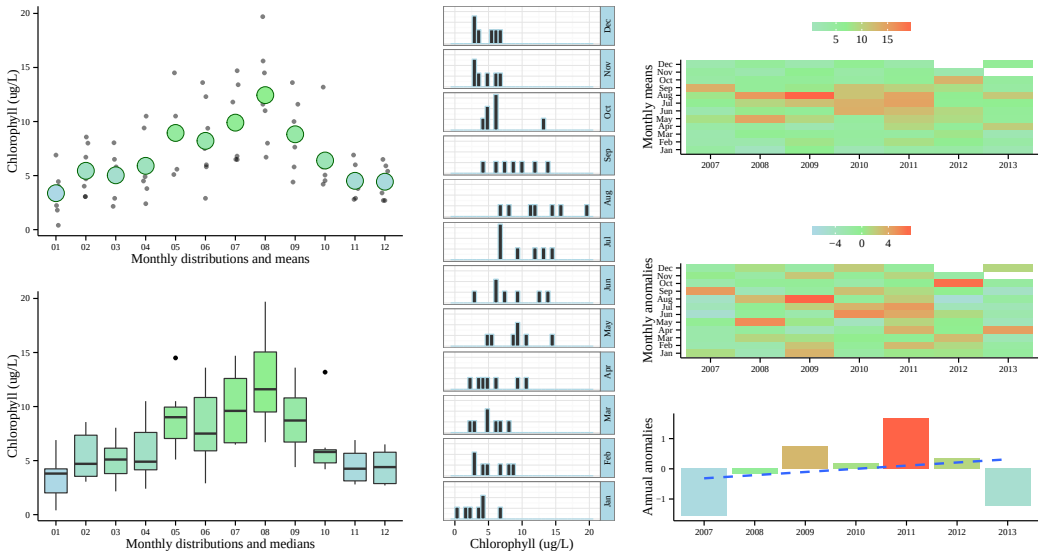

Figure 5: Summaries of a multi-year chlorophyll time series using the `plot_summary` function. Summaries include monthly distributions (means on top left, quantiles on bottom left), monthly histograms (center), monthly means by year (top right), deviation from monthly means (middle right), and annual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

```
> dat <- qaqc(apacpwq)
>
> ## plot
> overplot(dat, select = c('depth', 'do_mgl', 'ph', 'turb'),
+   subset = c('2013-01-01 0:0', '2013-02-01 0:0'), lwd = 2)
```

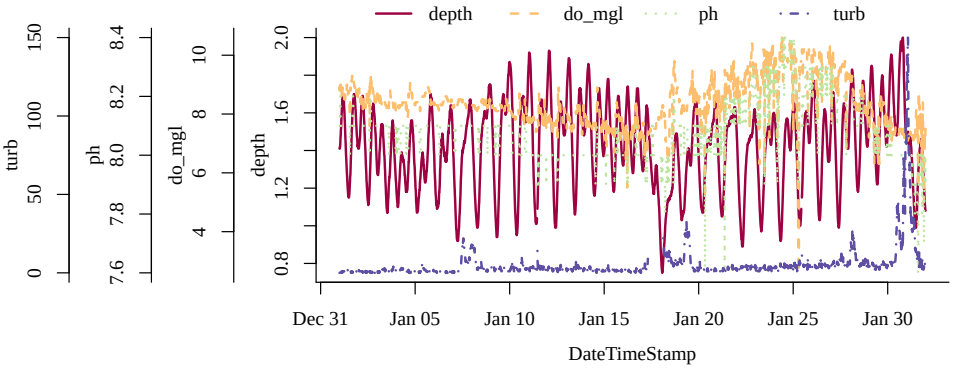

Figure 6: The `overplot` function plots multiple variables on the same y-axis.

Estimates of ecosystem metabolism provide a measure of overall system productivity to evaluate whether an ecosystem is a net source or sink of organic material. The open-water method (Odum, 1956) is a common approach to quantify 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 (Ro and Hunt, 2006; Thébault et al., 2008). A detailed discussion of the method is beyond the scope of this article, although users are strongly encouraged to consult references herein for additional information (see Kemp and Testa (2012); Needoba et al. (2012); Caffrey et al. (2013), also the package help files). The following is an example that shows use of the `ecometab` function from a combined water quality and weather data set. Monthly aggregations of the raw, daily estimates are plotted using `plot_metab` (Figure 7).

```
> ## 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)
```

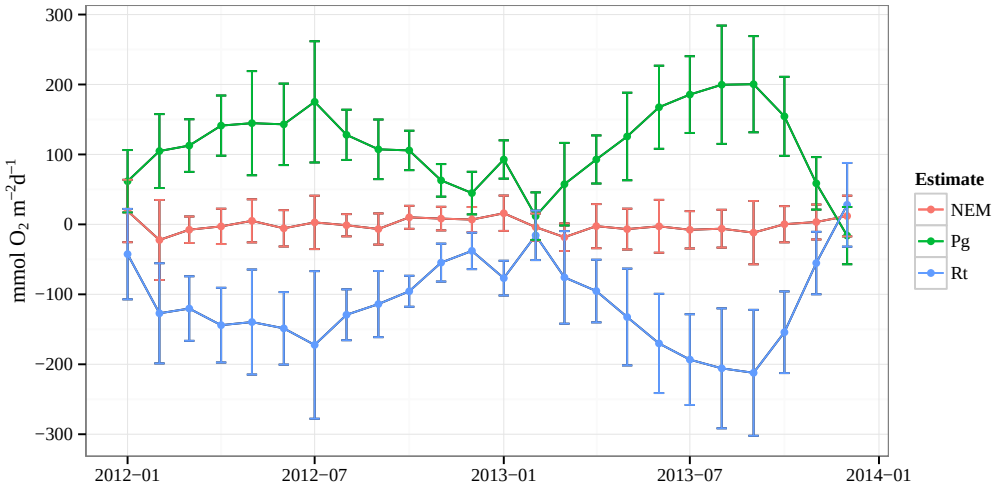

Figure 7: Monthly means (95% confidence) of ecosystem metabolism estimates (net ecosystem metabolism, gross production, and total respiration) for combined water quality and weather data for two years at Apalachicola Bay, Florida.

Finally, the `map_reserve` function can be used to create a map with all stations at a reserve using functions in the **ggmap** package (Kahle and Wickham, 2013). This map may be useful for aiding the interpretation of spatial trends in water quality parameters given the relative locations in a reserve. 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 can 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

Two **shiny** web applications hav been created that illustrate the improved ability to synthesize and evaluate multi-year time series using **SWMPr**. The first application evaluates trends in SWMP data within and between sites using an interactive **leaflet** map (Cheng and Xie (2015), Figure 9): https://beckmw.shinyapps.io/swmp_comp. Trends between reserves can be viewed using the map, whereas trends at individual sites can be viewed by clicking on a map location. Site-level trends are shown below the map with a simple linear regression to show an increase or decrease in values over time, whereas trends between sites are shown on the map for each station as circles that identify the direction and significance of the trend. The second application provides graphical summaries of water quality, weather, or nutrient station data at individual stations using the `plot_summary` function: https://beckmw.shinyapps.io/swmp_summary. The output is identical to Figure 5 with the addition of drop down menus to select the station, date range, and parameter for plotting.

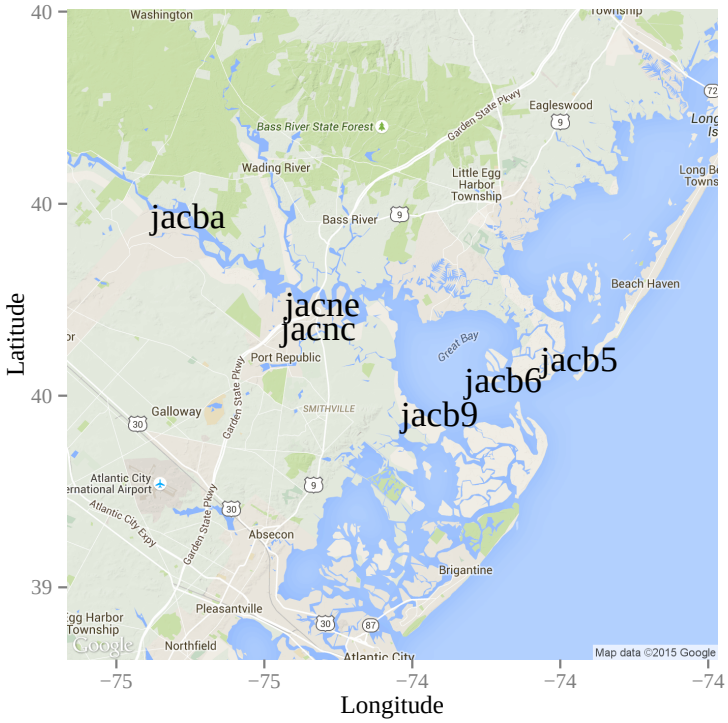

Figure 8: Locations of all sites at the Jacques Cousteau reserve using the `map_reserve` function.

Conclusions

SWMPr was developed to augment existing data management programs (i.e., CDMO) by providing a bridge betwen the raw data and the analysis software through its numerous data retrieval functions (Table 1). Established QAQC methods and data processing techniques are also enhanced with **SWMPr** by functions that filter observations for different QAQC flags (`qaqc`) and subset by selected dates or variables (`subset`). Additionally, challenges comparing differents datasets are addressed by the `setstep` and `comb` functions that standardize and combine time series. Finally, the analysis functions provide numerous tools to implement common analyses for time series and more specific methods for water quality data. Further development of the package will include modifications and additional functions to better integrate data analysis with the quality of information provided by SWMP. For example, several functions include default methods to extend use beyond the `"swmpr"` object and additional development will continue to focus on modifying the package to handle arbitrary data structures.

Acknowledgments

I acknowledge the significant work of NERRS researchers and staff that has allowed access to high-quality monitoring data. Thanks to Todd O'Brien for the inspiration for the online widgets. Thanks to Mike Murrell and Jim Hagy III for assistance with the ecosystem metabolism functions. Thanks to Jeffrey Hollister for providing useful comments on an earlier draft.

Bibliography

D. A. Bulthius. Distribution of seagrasses in a north Puget Sound estuary - Padilla Bay, Washington, USA. *Aquatic Botany*, 50(1):99–105, 1995. [p]

J. M. Caffrey. Production, respiration and net ecosystem metabolism in U.S. estuaries. *Environmental Monitoring and Assessment*, 81(1-3):207–219, 2003. [p]

J. M. Caffrey, M. C. Murrell, K. S. Amacker, J. Harper, S. Phipps, and M. Woodrey. 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*, 37(1):222–241, 2013. [p13]

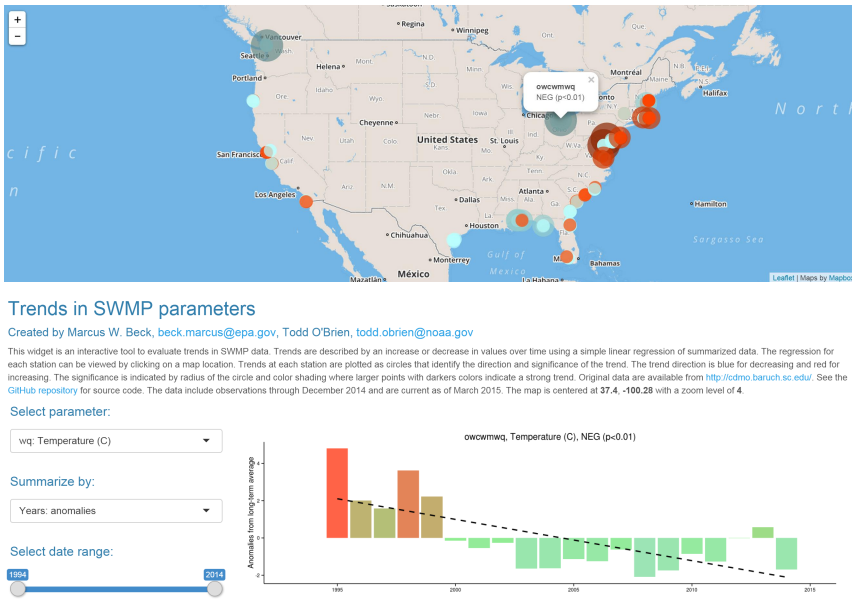

Figure 9: Online application for comparing trends in SWMP data parameters using an interactive map. Link: https://beckmw.shinyapps.io/swmp_comp

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2015. URL http://CRAN.R-project.org/package=shiny. R package version 0.11.1. [p12]

J. Cheng and Y. Xie. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*, 2015. URL http://CRAN.R-project.org/package=leaflet. R package version 1.0.0. [p14]

J. E. Cloern and A. D. Jassby. Patterns and scales of phytoplankton variability in estuarine-coastal ecosystems. *Estuaries and Coasts*, 33(2):230–241, 2010. [p11]

N. G. Dix, E. J. Phlips, and R. A. Gleeson. Water quality changes in the Guana Tolomato Matanzas National Estuarine Research Reserve, Florida, associated with four tropical storms. *Journal of Coastal Research*, 55(SI):26–37, 2008. [p]

M. Dowle, T. Short, S. Lianoglou, A. Srinivasan, R. Saporta, and E. Antonyan. *data.table: Extension of data.frame*, 2014. URL http://CRAN.R-project.org/package=data.table. R package version 1.9.4. [p8]

D. P. Fries, S. Z. Ivanov, P. H. Bhanushali, J. A. wilson, H. A. Broadbent, and A. C. Sanderson. Broadband, low-cost, coastal sensor nets. *Oceanography*, 20(4):150–155, 2008. [p1]

H. B. Glasgow, J. M. Burkholder, R. E. Reed, A. J. Lewitus, and J. E. Kleinman. 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*, 300(1-2):409–448, 2004. [p1]

A. D. Jassby and J. E. Cloern. *wq: Exploring water quality monitoring data*, 2014. URL http://CRAN.R-project.org/package=wq. R package version 0.4-1. [p8, 11]

D. Kahle and H. Wickham. ggmap: Spatial visualization with ggplot2. *The R Journal*, 5(1):144–161, 2013. URL http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf. [p8, 14]

W. M. Kemp and J. M. Testa. Metabolic balance between ecosystem production and consumption. In E. Wolanski and D. S. McLusky, editors, *Treatise on Estuarine and Coastal Science*, pages 83–118. Academic Press, New York, 2012. [p13]

M. Kendall and A. Stuart. *The Advanced Theory of Statistics*, volume 3. MacMillan Publishing Company, New York, New York, 1983. [p8, 10]

J. A. Needoba, T. D. Peterson, and K. S. Johnson. Method for the quantification of aquatic primary production and net ecosystem metabolism using in situ dissolved oxygen sensors. In S. M. Tiquia-Arashiro, editor, *Molecular Biological Technologies for Ocean Sensing*, pages 73–101. Springer, New York, 2012. [p13]

H. T. Odum. Primary production in flowing waters. *Limnology and Oceanography*, 1(2):102–117, 1956. [p13]

K. S. Ro and P. G. Hunt. 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, 2006. [p13]

D. M. Sanger, M. D. Arendt, Y. Chen, E. L. Wenner, A. F. Holland, D. Edwards, and J. Caffrey. A synthesis of water quality data: National estuarine research reserve system-wide monitoring program (1995-2000). Technical report, National Estuarine Research Reserve Technical Report Series 2002:3. South Carolina Department of Natural Resources, Marine Resources Division Contribution No. 500, Charleston, South Carolina, 2002. [p]

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/. [p1]

J. Thébault, T. S. Schraga, J. E. Cloern, and E. G. Dunlavey. Primary production and carrying capacity of former salt ponds after reconnection to San Francisco Bay. *Wetlands*, 28(3):841–851, 2008. [p13]

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

Marcus W Beck
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