Processing and organizing SWMP time series for analysis

Marcus W. Beck[1] Todd D. O'Brien[2]

[1]ORISE, USEPA NHEERL Gulf Ecology Division
Email: beck.marcus@epa.gov

[2]NOAA/NMFS Copepod Project
Email: todd.obrien@noaa.gov

Objectives and agenda

- Objectives
  - How can SWMP data quality be evaluated and handled?
  - How can data be selected and removed to facilitate analysis?
  - What are some ways that date are combined and why would this be done?

Objectives and agenda

- Objectives
  - How can SWMP data quality be evaluated and handled?
  - How can data be selected and removed to facilitate analysis?
  - What are some ways that date are combined and why would this be done?
- Agenda
  - Review of import and handling QAQC flags
  - Appropriate use of data subsets
  - Combining data for comparisons

Interactive portion

You can follow along in this module:

- dataset2
- script2

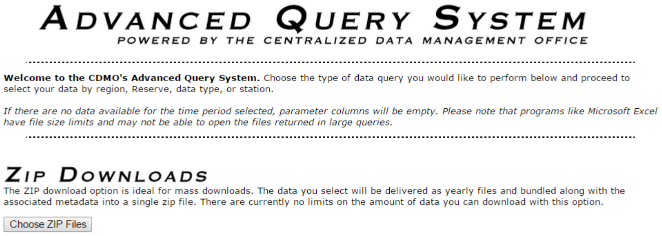
Interactive!

Retrieve SWMP data

We learned how to import SWMP data in the previous session

To review, the easiest approach is to download the data outside of R, then import using the ‘import˙local’ function

Be sure that you use only the zip downloads feature from CDMO - the ‘import˙local’ functions works best with these data


ADVANCED QUERY SYSTEM
POWERED BY THE CENTRALIZED DATA MANAGEMENT OFFICE

Welcome to the CDMO's Advanced Query System. Choose the type of data query you would like to perform below and proceed to select your data by region, Reserve, data type, or station.

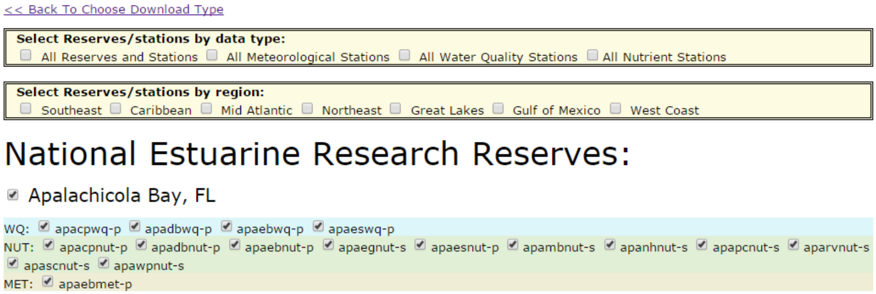
If there are no data available for the time period selected, parameter columns will be empty. Please note that programs like Microsoft Excel have file size limits and may not be able to open the files returned in large queries.

ZIP DOWNLOADS
The ZIP download option is ideal for mass downloads. The data you select will be delivered as yearly files and bundled along with the associated metadata into a single zip file. There are currently no limits on the amount of data you can download with this option.

Retrieve SWMP data

We have provided data for use with the workshop

For future access, it may be best to download all the data possible for a reserve to avoid repeated requests to the server and to centralize the location from which the data are imported into R


<< Back To Choose Download Type

Select Reserves/stations by data type:
☐ All Reserves and Stations ☐ All Meteorological Stations ☐ All Water Quality Stations ☐ All Nutrient Stations

Select Reserves/stations by region:
☐ Southeast ☐ Caribbean ☐ Mid Atlantic ☐ Northeast ☐ Great Lakes ☐ Gulf of Mexico ☐ West Coast

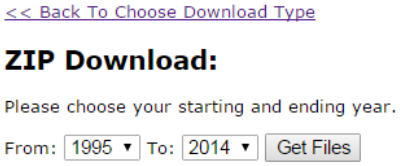
National Estuarine Research Reserves:

✓ Apalachicola Bay, FL

WQ: ✓ apacpwq-p ✓ apadbwq-p ✓ apaebwq-p ✓ apaeswq-p
NUT: ✓ apacpnut-p ✓ apadbnut-p ✓ apaebnut-p ✓ apaegnut-s ✓ apaesnut-p ✓ apambnut-s ✓ apanhnut-s ✓ apapcnut-s ✓ aparvnut-s ✓ apascnut-s ✓ apawpnut-s
MET: ✓ apaebmet-p

Retrieve SWMP data

It may be best to download all the data possible for a reserve to avoid repeated requests to the server and to centralize the location from which the data are imported into R



<< Back To Choose Download Type
ZIP Download:
Please choose your starting and ending year.
From: 1995 To: 2014 Get Files

Here we've made a request for all stations at Apalachicola Bay (water quality, nutrients, weather) and all available years (1995–2014)

This request will take several minutes to be delivered to your email - the files in 'dataset2' are an abbreviate version of these data for this training module

Retrieve SWMP data

Let's import some data for Apalachicola Bay

```
# reload the SWMPr package if you started a new session
library(SWMPr)

# import data
# change this path for the flash drive
path <- 'C:/data/dataset2'
wq_dat <- import_local(path, 'apacpwq')
nut_dat <- import_local(path, 'apacpnut')
met_dat <- import_local(path, 'apaebmet')
```

We've just imported data from 2011–2014 for three stations (apacpwq, apacpnut, apaebmet) and saved them in our workspace as three separate objects (wq_dat, nut_dat, met_dat)

Retrieve SWMP data

But don't take my word for it, take a look at the data!

```
# what are the dimenions of the water quality data?
dim(wq_dat)

## [1] 132035     25

# what are the dimenions of the nutrient data?
dim(nut_dat)

## [1] 48 13

# what are the dimenions of the weather data?
dim(met_dat)

## [1] 133548     23
```

Retrieve SWMP data
View the first six rows

```
# View the first six rows of the met data
head(met_dat)

##          datetimestamp atemp f_atemp rh f_rh   bp f_bp wspd f_wspd maxwspd
## 1 2011-01-01 00:00:00    15     <0> 94 <0> 1019 <0>     3    <0>       3
## 2 2011-01-01 00:15:00    15     <0> 95 <0> 1019 <0>     3    <0>       4
## 3 2011-01-01 00:30:00    15     <0> 95 <0> 1019 <0>     3    <0>       4
## 4 2011-01-01 00:45:00    15     <0> 95 <0> 1019 <0>     3    <0>       4
## 5 2011-01-01 01:00:00    15     <0> 95 <0> 1018 <0>     3    <0>       4
## 6 2011-01-01 01:15:00    15     <0> 95 <0> 1018 <0>     4    <0>       5
##   f_maxwspd wdir f_wdir sdwdir f_sdwdir totpar  f_totpar totprcp f_totprcp
## 1       <0>  145    <0>      8      <0>    0.8 <1> (CSM)       0       <0>
## 2       <0>  146    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
## 3       <0>  139    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
## 4       <0>  140    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
## 5       <0>  144    <0>      6      <0>    0.8 <1> (CSM)       0       <0>
## 6       <0>  141    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
##   cumprcp f_cumprcp totsorad f_totsorad
## 1       0       <0>       NA       <-1>
## 2       0       <0>       NA       <-1>
## 3       0       <0>       NA       <-1>
## 4       0       <0>       NA       <-1>
## 5       0       <0>       NA       <-1>
## 6       0       <0>       NA       <-1>
```

Retrieve SWMP data

View the last six rows

```
# View the last six rows of the met data
tail(met_dat)

##              datetimestamp atemp f_atemp rh f_rh   bp f_bp wspd f_wspd maxwspd
## 133543 2014-10-23 01:30:00    14     <0> 72 <0> 1017 <0>     3    <0>       5
## 133544 2014-10-23 01:45:00    14     <0> 72 <0> 1016 <0>     3    <0>       5
## 133545 2014-10-23 02:00:00    14     <0> 74 <0> 1016 <0>     3    <0>       4
## 133546 2014-10-23 02:15:00    14     <0> 74 <0> 1016 <0>     3    <0>       4
## 133547 2014-10-23 02:30:00    14     <0> 75 <0> 1016 <0>     3    <0>       4
## 133548 2014-10-23 02:45:00    14     <0> 76 <0> 1016 <0>     2    <0>       4
##        f_maxwspd wdir f_wdir sdwdir f_sdwdir totpar f_totpar totprcp f_totprcp
## 133543       <0>   33    <0>      9      <0>      0      <0>       0       <0>
## 133544       <0>   34    <0>     11      <0>      0      <0>       0       <0>
## 133545       <0>   36    <0>     10      <0>      0      <0>       0       <0>
## 133546       <0>   43    <0>     11      <0>      0      <0>       0       <0>
## 133547       <0>   41    <0>     10      <0>      0      <0>       0       <0>
## 133548       <0>   42    <0>     10      <0>      0      <0>       0       <0>
##        cumprcp f_cumprcp totsorad f_totsorad
## 133543      NA      <-2>       NA       <-1>
## 133544      NA      <-2>       NA       <-1>
## 133545      NA      <-2>       NA       <-1>
## 133546      NA      <-2>       NA       <-1>
## 133547      NA      <-2>       NA       <-1>
## 133548      NA      <-2>       NA       <-1>
```

Retrieve SWMP data

We'll first work with the water quality records

What class is the data?

```
# class of the data
class(met_dat)

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

This tells us that the data are two different classes - 'swmpr' and 'data.frame'

The class of an object is important because it defines the types of methods (i.e., functions) that apply

For example, 'head' and 'tail' functions work for a 'data.frame'

Retrieve SWMP data

The swmpr object class was developed to make your life easier working with SWMP data

The online documentation describes the functions that work with the swmpr object class, also...

```
# what functions/methods work with swmpr objects?
methods(class = 'swmpr')

##  [1] aggregate.swmpr comb.swmpr      decomp.swmpr    hist.swmpr
##  [5] lines.swmpr     na.approx.swmpr plot.swmpr      qaqc.swmpr
##  [9] qaqcchk.swmpr   setstep.swmpr   smoother.swmpr  subset.swmpr
```

Documentation of each function can be viewed as follows (although currently not complete):

```
# see help for a swmpr function
?aggregate.swmpr

# or...
help('aggregate.swmpr')
```

Retrieve SWMP data

A side note about R syntax... the convention ‘function.class’ means that a function applies to a specific class

The ‘function’ is generic, whereas the ‘function.class’ is a method for a class that applies to the generic

```
# view the methods that apply to the generic aggregate
methods('aggregate')

## [1] aggregate.data.frame aggregate.default*   aggregate.formula*
## [4] aggregate.swmpr      aggregate.ts         aggregate.zoo*
##
##    Non-visible functions are asterisked
```

A function with a class method can be executed using shorthand...

```
# shorthand for executing aggregate on a swmpr object
aggregate(met_dat, by = 'quarters')

# long also works
aggregate.swmpr(met_dat, by = 'quarters')
```

Retrieve SWMP data

A useful feature of R is that a class will have both ***data*** and ***attributes***

For the swmpr class, the ***data*** are the raw swmpr data as a data.frame

The ***attributes*** are a list of metadata for the imported data

```
# what attributes are available for a swmpr object
names(attributes(met_dat))

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

# view the parameters
attr(met_dat, 'parameters')

##  [1] "atemp"    "rh"       "bp"       "wspd"     "maxwspd"  "wdir"
##  [7] "sdwdir"   "totpar"   "totprcp"  "cumprcp"  "totsorad"
```

Retrieve SWMP data

You can also view all the attributes as follows:

```
# view all attributes
attributes(met_dat)
```

This is not recommended since they are quite long, e.g., an attribute of the 'data.frame' class is the row names (132035 rows for'wq_dat')

Individual attributes are useful for getting a feel for the dataset - what is the date range? what parameters are included? are QAQC columns present?

However, the intended use of attributes is behind the scenes with swmpr functions - they will be used to process the data and updated automatically

Retrieve SWMP data

A summary of the swmpr object class:

- Throughout ‘SWMPr’ refers to the ***package***, ‘swmpr’ refers to the object ***class***
- ***Methods*** aka functions in the SWMPr package are specific for swmpr objects - see the help documentation (‘?aggregate.swmpr’)
- The swmpr object has both ***data*** and ***attributes*** - the data are in the ‘data.frame’ format, the attributes are in a ‘list’

These are basic concepts that are fundamental to how the R language works – you should have a general understanding of their meaning

Organize SWMP data

Now that we have a feel for the data, what needs to be done before we can start analyzing the information?

Last module:

- How do we handle QAQC data or ‘bad’ observations?
- How do we deal with data we don’t want?
- How do we combine data for comparison?
- How do we handle issues inherent with time series?

Several of these problems are context-dependent - driven by the question or analysis

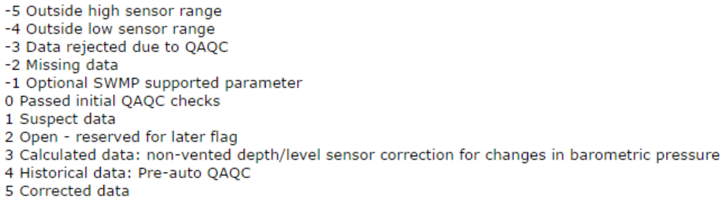
Others are common to any analysis...

Organize SWMP data

Perhaps the first organizational tool you want to use is ‘qaqc.swmpr’

This function does two things:

- Remove observations with a specified QAQC flag value
- Remove QAQC columns



-5 Outside high sensor range
-4 Outside low sensor range
-3 Data rejected due to QAQC
-2 Missing data
-1 Optional SWMP supported parameter
0 Passed initial QAQC checks
1 Suspect data
2 Open - reserved for later flag
3 Calculated data: non-vented depth/level sensor correction for changes in barometric pressure
4 Historical data: Pre-auto QAQC
5 Corrected data

Retrieve SWMP data

Remember, each parameter has a QAQC column preceded by 'f''

```
# View the first six rows of the met data
head(met_dat)

##          datetimestamp atemp f_atemp rh f_rh   bp f_bp wspd f_wspd maxwspd
## 1 2011-01-01 00:00:00    15     <0> 94  <0> 1019  <0>    3    <0>       3
## 2 2011-01-01 00:15:00    15     <0> 95  <0> 1019  <0>    3    <0>       4
## 3 2011-01-01 00:30:00    15     <0> 95  <0> 1019  <0>    3    <0>       4
## 4 2011-01-01 00:45:00    15     <0> 95  <0> 1019  <0>    3    <0>       4
## 5 2011-01-01 01:00:00    15     <0> 95  <0> 1018  <0>    3    <0>       4
## 6 2011-01-01 01:15:00    15     <0> 95  <0> 1018  <0>    4    <0>       5
##   f_maxwspd wdir f_wdir sdwdir f_sdwdir totpar  f_totpar totprcp f_totprcp
## 1       <0>  145    <0>      8      <0>    0.8 <1> (CSM)       0       <0>
## 2       <0>  146    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
## 3       <0>  139    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
## 4       <0>  140    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
## 5       <0>  144    <0>      6      <0>    0.8 <1> (CSM)       0       <0>
## 6       <0>  141    <0>      7      <0>    0.8 <1> (CSM)       0       <0>
##   cumprcp f_cumprcp totsorad f_totsorad
## 1       0       <0>       NA       <-1>
## 2       0       <0>       NA       <-1>
## 3       0       <0>       NA       <-1>
## 4       0       <0>       NA       <-1>
## 5       0       <0>       NA       <-1>
## 6       0       <0>       NA       <-1>
```

Organize SWMP data

You will have to decide which values to keep - be conservative and only keep those that passed QAQC or keep all the data

To help you decide, it may be useful to get an idea of the distribution of QAQC flags in the data, use ‘qaqcchk’

```
# use qaqcchk to view distribution of qaqc flags
myqaqc <- qaqcchk(met_dat)
```

This function returns a data.frame

- The first column shows all the QAQC codes in the data
- The remaining columns show the counts for each parameter of the observations assigned to each QAQC code

Organize SWMP data

```
# a subset of results from the qaqcchk function
head(myqaqc)

##                piece f_atemp f_bp f_cumprcp f_maxwspd f_rh f_sdwdir f_totpar
## 1        <-2> [GPD]        2    2         2         2    2        2        2
## 2        <-3> [GMT]        5   13        70        16    5       16       18
## 3        <-3> [GPD]       15   16        16        16   15       16       16
## 4        <-3> [GPR]       14   14        14        14   14       14       13
## 5        <-3> [SMT]        2   NA       121         3    2        3        2
## 6 <-3> [SQR] (CSM)        3   NA        NA        NA    3       NA     4023
##   f_totprcp f_totsorad f_wdir f_wspd
## 1         2         NA      2      2
## 2        13         NA     16     16
## 3        16         NA     16     16
## 4        14         NA     14     14
## 5        14         NA      3      3
## 6        NA         NA     NA     NA

# or view all in a separate window
View(myqaqc)
```

Link to QAQC codes

Organize SWMP data

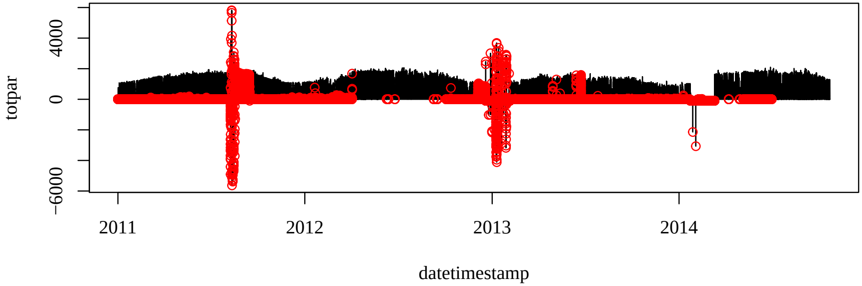
A plot of the data may also be useful to view QAQC flags, but this is tedious

Here's an example for the 'totpar' variable

```
# select values that did not pass qaqc
nopass <- grep('0', met_dat$f_totpar, invert = T)
nopass <- met_dat[nopass, ]

# plot totpar from met_dat
plot(totpar ~ datetimestamp, met_dat, type = 'l')

# add  points that did not pass qaqc
points(nopass$datetimestamp, nopass$totpar, col = 'red')
```

Organize SWMP data

Observations in red are those that did not pass QAQC checks - some are obvious, others are not



Does this plot make sense??

Organize SWMP data

You should have an idea of how you want to handle QAQC values after viewing the output from qaqcchk or plotting - or you already knew

Next, use the ‘qaqc’ function...

```
# filter observations by qaqc flags, remove qaqc columns
met_qaqc <- qaqc(met_dat)
```

The default behavior for this function is to keep only observations with a ‘0’ QAQC flag - data that passed initial checks

See the help documentation for the function

```
# view help file
?qaqc
```

Organize SWMP data

View the data after keeping only values that passed QAQC ('0' flag)

```
# data after qaqc processing
head(met_qaqc)

##          datetimestamp atemp rh   bp wspd maxwspd wdir sdwdir totpar totprcp
## 1 2011-01-01 00:00:00    15 94 1019    3       3  145      8     NA       0
## 2 2011-01-01 00:15:00    15 95 1019    3       4  146      7     NA       0
## 3 2011-01-01 00:30:00    15 95 1019    3       4  139      7     NA       0
## 4 2011-01-01 00:45:00    15 95 1019    3       4  140      7     NA       0
## 5 2011-01-01 01:00:00    15 95 1018    3       4  144      6     NA       0
## 6 2011-01-01 01:15:00    15 95 1018    4       5  141      7     NA       0
##   cumprcp totsorad
## 1       0       NA
## 2       0       NA
## 3       0       NA
## 4       0       NA
## 5       0       NA
## 6       0       NA
```

Organize SWMP data

What if we want to keep all the values, regardless of flag?

```
# keep all values
met_qaqc <- qaqc(met_dat, qaqc_keep = NULL)

head(met_qaqc) # note the totpar column compared to the last example


##           datetimestamp atemp rh   bp wspd maxwspd wdir sdwdir totpar totprcp
## 1 2011-01-01 00:00:00    15 94 1019    3       3  145      8    0.8       0
## 2 2011-01-01 00:15:00    15 95 1019    3       4  146      7    0.8       0
## 3 2011-01-01 00:30:00    15 95 1019    3       4  139      7    0.8       0
## 4 2011-01-01 00:45:00    15 95 1019    3       4  140      7    0.8       0
## 5 2011-01-01 01:00:00    15 95 1018    3       4  144      6    0.8       0
## 6 2011-01-01 01:15:00    15 95 1018    4       5  141      7    0.8       0
##   cumprcp totsorad
## 1       0       NA
## 2       0       NA
## 3       0       NA
## 4       0       NA
## 5       0       NA
## 6       0       NA
```

Organize SWMP data

If you're not convinced, try removing only the '0' flag

```
# keep all values
to_keep <- c(-5, -4, -3, -2, -1, 1, 2, 3, 4, 5)
met_qaqc <- qaqc(met_dat, qaqc_keep = to_keep)

# does this result make sense??
head(met_qaqc)

##          datetimestamp atemp rh bp wspd maxwspd wdir sdwdir totpar totprcp
## 1 2011-01-01 00:00:00    NA NA NA   NA      NA   NA     NA    0.8      NA
## 2 2011-01-01 00:15:00    NA NA NA   NA      NA   NA     NA    0.8      NA
## 3 2011-01-01 00:30:00    NA NA NA   NA      NA   NA     NA    0.8      NA
## 4 2011-01-01 00:45:00    NA NA NA   NA      NA   NA     NA    0.8      NA
## 5 2011-01-01 01:00:00    NA NA NA   NA      NA   NA     NA    0.8      NA
## 6 2011-01-01 01:15:00    NA NA NA   NA      NA   NA     NA    0.8      NA
##   cumprcp totsorad
## 1      NA       NA
## 2      NA       NA
## 3      NA       NA
## 4      NA       NA
## 5      NA       NA
## 6      NA       NA
```

Organize SWMP data

We'll continue by using values that passed the QAQC checks

```
# continue with qaqc processed data

# water quality
# note the column number before/after qaqc processing
dim(wq_dat)

## [1] 132035     25

wq_dat <- qaqc(wq_dat)
dim(wq_dat)

## [1] 132035     13

# nutrients
nut_dat <- qaqc(nut_dat)

# weather
met_dat <- qaqc(met_dat)
```

Organize SWMP data

What is the next logical step after dealing with QAQC values?

How would we further want to organize the data?

Maybe we want to subset the data...

For example, we don't want all the data columns or we only want to work with a specific date range

Use the subset function...

```
# view help file
?subset.swmpr
```

Note that R has a generic subset function, subset.swmpr is a method for swmpr objects

Organize SWMP data

The subset.swmpr function has several arguments

```
# view the arguments for subset.swmpr
formals(subset.swmpr)

## $swmpr_in
##
##
## $subset
## NULL
##
## $select
## NULL
##
## $operator
## NULL
##
## $rem_rows
## F
##
## $rem_cols
## F
```

Organize SWMP data

The subset.swmpr function has several arguments

- swmpr_in: input data (swmpr object)
- subset: dates to keep
- select: parameters to keep
- operator: less than, greater than if only one date in subset
- rem_rows: remove empty rows
- rem_cols: remove empty columns

We'll go through examples that use the function with different arguments

Organize SWMP data

The simplest use of ‘subset.swmpr’ is to remove empty rows and columns – this is typically not a major issue but it’s convenient if you’re a compulsive data cleaner

```
# rows, columns in wq_dat
dim(wq_dat)

## [1] 132035     13

# remove empty rows, columns
tmp <- subset(wq_dat, rem_rows = T, rem_cols = T)

# dimensions after removing empty rows, columns
dim(tmp)

## [1] 124273      9
```

About 1000 rows and four columns of missing data!

Organize SWMP data

The ‘select’ argument of ‘subset.swmpr’ is used to select parameters of interest - one to many

```
# select the DO column
tmp <- subset(wq_dat, select = 'do_mgl')
head(tmp)

##          datetimestamp do_mgl
## 1 2011-01-01 00:00:00      6
## 2 2011-01-01 00:15:00      6
## 3 2011-01-01 00:30:00      6
## 4 2011-01-01 00:45:00      6
## 5 2011-01-01 01:00:00      6
## 6 2011-01-01 01:15:00      6
```

Note that the datetimestamp is retained, this column will always be included in a subset

Organize SWMP data

The ‘select’ argument of ‘subset.swmpr’ is used to select parameters of interest - one to many

```
# select DO and salinity
tmp <- subset(wq_dat, select = c('do_mgl', 'sal'))
head(tmp)

##          datetimestamp sal do_mgl
## 1 2011-01-01 00:00:00  28      6
## 2 2011-01-01 00:15:00  28      6
## 3 2011-01-01 00:30:00  28      6
## 4 2011-01-01 00:45:00  28      6
## 5 2011-01-01 01:00:00  29      6
## 6 2011-01-01 01:15:00  29      6
```

Note the use of the concatenate function ‘c’

Try this...

```
# incorrect syntax
subset(wq_dat, select = 'do_mgl', 'sal')

## Error:  subset must be of format %Y-%m-%d %H:%M
```

Organize SWMP data

The ‘subset’ argument of ‘subset.swmpr’ selects a date range

The dates must have a specific format - ‘YYYY-mm-dd HH:MM’

```
# select a date range, July 2012
dates <- c('2012-07-01 12:00', '2012-07-31 6:30')
tmp <- subset(wq_dat, subset = dates)
head(tmp) # view first six rows
```

```
##          datetimestamp temp spcond sal do_pct do_mgl depth cdepth level clevel
## 1 2012-07-01 12:00:00   29     34  21     86      6     2     NA    NA     NA
## 2 2012-07-01 12:15:00   28     34  21     83      6     2     NA    NA     NA
## 3 2012-07-01 12:30:00   28     34  21     74      5     2     NA    NA     NA
## 4 2012-07-01 12:45:00   28     34  21     74      5     2     NA    NA     NA
## 5 2012-07-01 13:00:00   28     34  21     73      5     2     NA    NA     NA
## 6 2012-07-01 13:15:00   28     34  22     70      5     2     NA    NA     NA
##   ph turb chlfluor
## 1  8    7       NA
## 2  8    6       NA
## 3  8    5       NA
## 4  8    6       NA
## 5  8    7       NA
## 6  8    8       NA
```

Organize SWMP data

You can also verify the subset by checking the attributes of the swmpr object

```
# check the date_rng attribute
class(tmp) # a swmpr object?

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

# what are the attributes?
names(attributes(tmp))

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

# get the date range
attr(tmp, 'date_rng')

## [1] "2012-07-01 12:00:00 EST" "2012-07-31 06:30:00 EST"
```

Organize SWMP data

Observations earlier or later than a date can also be selected

This also requires the 'operator' argument – $>$, $<$, $>=$, $<=$, $==$, $!=$

```
# get observations for 2013
dates <- '2013-01-01 00:00'
tmp <- subset(wq_dat, subset = dates, operator = '>=')
head(tmp)
```

```
##          datetimestamp temp spcond sal do_pct do_mgl depth cdepth level clevel
## 1 2013-01-01 00:00:00   13     42  27     99      9     1     NA    NA     NA
## 2 2013-01-01 00:15:00   13     42  27     99      9     1     NA    NA     NA
## 3 2013-01-01 00:30:00   13     42  27    101      9     1     NA    NA     NA
## 4 2013-01-01 00:45:00   13     42  27    102      9     1     NA    NA     NA
## 5 2013-01-01 01:00:00   13     42  27    100      9     1     NA    NA     NA
## 6 2013-01-01 01:15:00   13     42  27    101      9     1     NA    NA     NA
##   ph turb chlfluor
## 1  8    0       NA
## 2  8    1       NA
## 3  8    1       NA
## 4  8    1       NA
## 5  8    0       NA
## 6  8    0       NA
```

Organize SWMP data

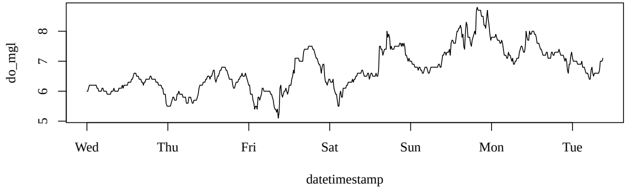
Try a simple application of ‘subset’ - plot dissolved oxygen and water temperature for October 2014

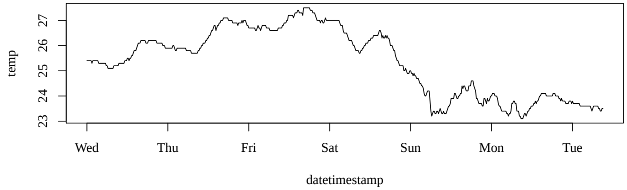
```
# dates and parameters  to select
dates <- '2014-10-01 00:00'
params <- c('do_mgl', 'temp')

# subset
tmp <- subset(wq_dat, select = params, subset = dates, operator = '>=')
head(tmp)

##          datetimestamp temp do_mgl
## 1 2014-10-01 00:00:00   25      6
## 2 2014-10-01 00:15:00   25      6
## 3 2014-10-01 00:30:00   25      6
## 4 2014-10-01 00:45:00   25      6
## 5 2014-10-01 01:00:00   25      6
## 6 2014-10-01 01:15:00   25      6
```

Organize SWMP data

Try a simple application of 'subset' - plot dissolved oxygen and water temperature for October 2014

```r
# plot DO and water temp
plot(do_mgl ~ datetimestamp, data = tmp, type = 'l')
plot(temp ~ datetimestamp, data = tmp, type = 'l')
```





Organize SWMP data

Common problems with ‘subset.swmpr’...

```
# incorrect subset argument format, no time
subset(wq_dat, subset = c('2012-01-01', '2012-01-31'))

## Error:  subset must be of format %Y-%m-%d %H:%M

# forgot to include operator
subset(wq_dat, subset = '2012-01-31 00:00')

## Error:  Binary operator must be included if only one subset value is provided

# incorrect parameter names
subset(wq_dat, select = 'DO')

## Error:  select argument is invalid
```

How do we fix these problems??

Organize SWMP data

Now that we know how to handle QAQC flags and subset the data, what else could we do before we analyze?

What if we want to compare time series from different datasets?

Data are imported separately for water quality, weather, and nutrients...

Sometimes the time step is different...

Use the ‘comb.swmp’ and ‘setstep.swmp’ functions!

```
# help files
?comb.swmpr
?setstep.swmpr
```

Organize SWMP data

The ‘setstep.swmpr’ function is used to standardize the time step of a swmpr object

The ‘comb.swmpr’ function is used to combine swmpr objects

‘setstep.swmpr’ is used within ‘comb.swmpr’ so you should not have to use it directly

How is it done??

Organize SWMP data

How can we combine SWMP data?

```
# combine water quality and weather data in the same object
tmp <- comb(wq_dat, met_dat)
head(tmp, 3) # first three rows
```

```
##          datetimestamp atemp rh   bp wspd maxwspd wdir sdwdir totpar totprcp
## 1 2011-01-01 00:00:00    15 94 1019    3       3  145      8     NA       0
## 2 2011-01-01 00:15:00    15 95 1019    3       4  146      7     NA       0
## 3 2011-01-01 00:30:00    15 95 1019    3       4  139      7     NA       0
##   cumprcp totsorad temp spcond sal do_pct do_mgl depth cdepth level clevel ph
## 1       0       NA   11     44  28     68      6     2     NA    NA     NA  8
## 2       0       NA   11     44  28     68      6     2     NA    NA     NA  8
## 3       0       NA   11     44  28     68      6     2     NA    NA     NA  8
##   turb chlfluor
## 1    3       NA
## 2    3       NA
## 3    2       NA
```

Organize SWMP data

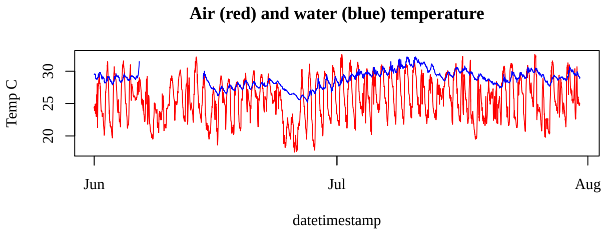
What happened? Check the attributes of the combined object...

```
# attributes of combined object

# stations
attr(tmp, 'station')

## [1] "apacpwq"  "apaebmet"

# date ranges, same as before
attr(tmp, 'date_rng')

## [1] "2011-01-01 00:00:00 EST" "2014-10-23 02:45:00 EST"

# parameters, both wq and met
attr(tmp, 'parameters')

##  [1] "atemp"    "rh"       "bp"       "wspd"     "maxwspd"  "wdir"
##  [7] "sdwdir"   "totpar"   "totprcp"  "cumprcp"  "totsorad" "temp"
## [13] "spcond"   "sal"      "do_pct"   "do_mgl"   "depth"    "cdepth"
## [19] "level"    "clevel"   "ph"       "turb"     "chlfluor"
```

Organize SWMP data

We now have a swmpr object with data from two stations, why do we want this? Easier plotting...

```
# plot some combined data

# subset date ranges first
dates <- c('2012-06-01 0:0', '2012-07-31 0:0')
to_plot <- subset(tmp, subset = dates)

# plot
plot(atemp ~ datetimestamp, to_plot, type = 'l', col = 'red', ylab = 'Temp C')
lines(to_plot$datetimestamp, to_plot$temp, col = 'blue')
title('Air (red) and water (blue) temperature')
```



Organize SWMP data

The options for ‘comb.swmpr’ allow flexibility

```
# arguments for the function, or ?comb.swmpr
formals(comb.swmpr)

## $...
##
##
## $timestep
## [1] 15
##
## $differ
## [1] 5
##
## $method
## [1] "union"
```

Organize SWMP data

What do the arguments specify?

- ... : input swmpr data, separated by comma
- timestep: minutes defining the standardized time step
- differ: maximum difference in minutes for matching observations with original time steps to standardized time steps
‘
- method: how the data are combined using the time stamps - union, intersect, or using a station

Organize SWMP data

Changing the ‘timestep’ argument can be useful for reducing data volume...

```
# dimension of earlier combined object
dim(tmp)

## [1] 133548     24

# create new object at two hour time step
tmp <- comb(wq_dat, met_dat, timestep = 120)
dim(tmp)

## [1] 16695    24
```

Organize SWMP data

Changing the ‘timestep’ argument can be useful for reducing data volume...

```
# note the time step in datetimestamp
head(tmp, 4)


##          datetimestamp atemp rh   bp wspd maxwspd wdir sdwdir totpar totprcp
## 1 2010-12-31 23:00:00    NA NA   NA   NA      NA   NA     NA     NA      NA
## 2 2011-01-01 01:00:00    15 95 1018    3       4  144      6     NA       0
## 3 2011-01-01 03:00:00    15 93 1017    6       7  137      7     NA       0
## 4 2011-01-01 05:00:00    15 95 1017    4       6  146      8     NA       0
##   cumprcp totsorad temp spcond sal do_pct do_mgl depth cdepth level clevel ph
## 1      NA       NA   NA     NA  NA     NA     NA    NA     NA    NA     NA NA
## 2       0       NA   11     44  29     68      6     2     NA    NA     NA  8
## 3       0       NA   11     45  29     66      6     2     NA    NA     NA  8
## 4       0       NA   11     44  28     66      6     1     NA    NA     NA  8
##   turb chlfluor
## 1   NA       NA
## 2    2       NA
## 3    3       NA
## 4    2       NA
```

Organize SWMP data

Caution! Standardizing a time series to a set time step may impose some inaccuracy

Observations are matched as close in time as possible to the standardized time series

For example, if we set the time step at 60 minute intervals and actual observations are taken every 15 and 45 minutes of the hour

Which observations are matched and which are discarded?

By default, 'comb.swmpr' matches the closest observation and discards the rest

'Close' is defined by the 'differ' argument

Organize SWMP data

The ‘differ’ argument defines the maximum time difference for matching observations to the standardized time step

The maximum allowed value for ‘differ’ is one half the time step – values beyond this window will create duplicates in your data

For example, set a 60 minute time step – 5pm, 6pm, 7pm, etc., consider an observation at 635pm

wrong!

- ‘differ’ is 40 minutes
- observation will be duplicated at 6pm and 7pm

right!

- ‘differ’ is 30 minutes
- observation is matched only to 7pm

Again, default behavior of ‘comb.swmpr’ prevents duplication

Organize SWMP data

Consider the alternative scenario of a using a conservative value for 'differ'

For example, set a 60 minute time step – 5pm, 6pm, 7pm, etc., consider an observation at 635pm

If 'differ' is five minutes, there will be no data in the standardized time series!

Fortunately, most SWMP data already follow a standard 15 minute time step

```
# note the pre-existing time steps
head(wq_dat$datetimestamp)

## [1] "2011-01-01 00:00:00 EST" "2011-01-01 00:15:00 EST"
## [3] "2011-01-01 00:30:00 EST" "2011-01-01 00:45:00 EST"
## [5] "2011-01-01 01:00:00 EST" "2011-01-01 01:15:00 EST"
```

Organize SWMP data

However, nutrient data may have irregular time stamps since these data are not from 'continuous' monitoring station

```
# note irregular time stamps on nutrient data
head(nut_dat)


##         datetimestamp  po4f nh4f no2f no3f no23f chla_n
## 1 2011-01-11 11:28:00    NA 0.04   NA   NA    NA      4
## 2 2011-02-09 12:51:00 0.005   NA   NA   NA  0.09      8
## 3 2011-02-09 12:55:00 0.004   NA   NA   NA  0.10      9
## 4 2011-02-09 13:03:00 0.004   NA   NA   NA  0.07      8
## 5 2011-03-02 10:36:00 0.004   NA   NA   NA  0.05      5
## 6 2011-04-06 11:29:00    NA 0.04   NA   NA  0.24      9
```

Use caution if combining these data or standardizing the time step - there is potential for data omission or extrapolation

Organize SWMP data

Case in point, try standardizing the nutrient time series with 'setstep.swmpr'

```
dim(nut_dat) # initial dimensions

## [1] 48  7

# standardize nutrient time series
tmp <- setstep(nut_dat, timestep = 60, differ = 5)
dim(tmp)

## [1] 25369     7

# remove empty rows, columns
tmp <- subset(tmp, rem_row = T, rem_col = T)
tmp #only four rows!

##          datetimestamp  po4f nh4f no23f chla_n
## 1 2011-02-09 13:00:00 0.004   NA 0.075      8
## 2 2012-07-03 10:00:00 0.004 0.07 0.057      6
## 3 2012-09-05 11:00:00    NA 0.03 0.007      9
## 4 2013-07-09 09:00:00 0.004 0.06 0.036      7
```

Organize SWMP data

A final note about combining data... what about combining data with different time ranges

Consider combining two datasets

- Scenario 1: Time ranges are the same
- Scenario 2: Time ranges are not the same, but there is overlap
- Scenario 3: Time ranges are not the same, there is no overlap

The ‘method’ argument of ‘comb.swmpr’ allows flexibility under different scenarios - time range intersect, union, or range of one station

Organize SWMP data

Scenario 1: Time ranges are the same

Don't worry about changing the 'method' argument - it will have no effect

Our previous examples represent scenario 1

Organize SWMP data

Scenario 1: Time ranges are the same

Don't worry about changing the 'method' argument - it will have no effect

Our previous examples represent scenario 1

Scenario 2: Time ranges are not the same, but there is overlap

- union: the default method, use the whole time range of both datasets
- intersect: use only the time range that is shared between the two
- or enter one of the station names, this says use the time range belonging to that station

Organize SWMP data

Scenario 1: Time ranges are the same

Don't worry about changing the 'method' argument - it will have no effect

Our previous examples represent scenario 1

Scenario 2: Time ranges are not the same, but there is overlap

- union: the default method, use the whole time range of both datasets
- intersect: use only the time range that is shared between the two
- or enter one of the station names, this says use the time range belonging to that station

Scenario 3: Time ranges are not the same, there is no overlap

The only method that makes sense is 'union'

Organize SWMP data

Try some examples

```
# first we subset for the example
sub1 <- subset(wq_dat, subset = c('2012-07-01 0:0', '2012-07-31 0:0'))
sub2 <- subset(met_dat, subset = c('2012-07-15 0:0', '2012-08-12 0:0'))

# what are the date ranges?
attr(sub1, 'date_rng')

## [1] "2012-07-01 EST" "2012-07-31 EST"

attr(sub2, 'date_rng')

## [1] "2012-07-15 EST" "2012-08-12 EST"
```

Organize SWMP data

Try some examples

```
# combine using union, this is the default
tmp <- comb(sub1, sub2, method = 'union')

# what is the date range of the combined object?
attr(tmp, 'date_rng')

## [1] "2012-07-01 EST" "2012-08-12 EST"
```

Organize SWMP data

Try some examples

```
# combine using intersect
tmp <- comb(sub1, sub2, method = 'intersect')

# what is the date range of the combined object?
attr(tmp, 'date_rng')

## [1] "2012-07-15 EST" "2012-07-31 EST"
```

Organize SWMP data

Try some examples

```
# combine using the time range in sub1
station <- attr(sub1, 'station')
tmp <- comb(sub1, sub2, method = station)

# what is the date range of the combined object?
attr(tmp, 'date_rng')

## [1] "2012-07-01 EST" "2012-07-31 EST"
```

Organize SWMP data

Now you have an idea of how to organize SWMP data for analysis!

Here's what we did:

- Evaluate QAQC flags in the data and handle accordingly
- Subsetting the data to remove empty rows/columns or to select variables or time ranges of interest
- Combining data for comparison or data simplification

Consult the SWMPr cookboook for an example workflow!

After lunch... what are some basic ways we can analyze the data?

NERRS / SWMP
Data Analysis Workshop: *Time Series*
November 17, 2014

Questions??