The DataflowR Package for Processing Underway Water Quality Surveys

Joseph Stachelek

February 15, 2017

Contents

1 Introduction

The `DataflowR` package is an integrated set of processing functions to process underway water quality data. These functions provide the ability to perform quality assurance checks, exploratory analysis, spatial interpolation, and data management operations. Functions to handle associated discrete grab samples are also included.

2 Programs and Applications

- R (required; must have the `DataflowR` package and its dependencies installed)

- RStudio (optional; for more easily accessing documentation)

- GRASS (optional; alternatively use QGIS, ArcGIS, Surfer, etc, for more easily adjusting final symbology)

3 Installing the `DataflowR` package

3.0.1 Pre-built Installation

The `R` package `DataflowR` is distributed via a `.tar.gz` (analagous to `.zip`) package archive file. This package contains the source code for package functions as well as the archived datasets for the SFWMD Florida Bay Dataflow Monitoring Program. In RStudio, it can be installed by navigating to `Tools -> Install Packages...` `-> Install from: -> Package Archive File`. Computers running the Windows operating system can only install binary `.zip` package archive files unless they have additional compiler software installed (RTools). The `DataflowR` binary package can be installed by running the following command from the `R` console:

```
> install.packages(path.to.zip, type = "win.binary", repos = NULL,
+                   dependencies = TRUE)
```

where `path.to.zip` is replaced by the file path of the `.zip` file.

3.0.2 Source Installation

The `DataflowR` package can also be built directly from source code if no pre-built `.tar.gz` (`.zip`) package is available. This can be accomplished by installing the `devtools` package and running the following set of commands:

```
> install.packages("devtools")
> devtools::install_github("jsta/DataflowR")
```

where `<username>` and `<password>` are replaced with your GitLab username and password. On Windows machines, the `RTools` program is required for source installation.

3.1   Data Archive

Archived Dataflow output is distributed seperately from the `DataflowR` package as a compressed file directory.

☞ IMPORTANT!! before continuing, copy the contents of the Dataflow archive to a local folder such as: `C:\Documents\Data\Dataflow`.
This will become your "working directory". Package functions will read and write to this directory without directly modifying archived datasets.
For example, the structure of `C:\Documents\Data\Dataflow` would look like:

```
 Dataflow
 DF_BaseFile
 DF_FullDataSets
 DF_Subsets
 DF_Surfaces
 DF_Validation
```

Next, load the `DataflowR` package to discover the location of the `localpath` file. Alternatively, you can discover the path to the file by running the following command:

```
> system.file("localpath", package = "DataflowR")
```

Update the first line of the `localpath` file to point to the location of your local copy of the Data Archive folder. End the file with a blank line. Note that if you are on a Windows machine you need to have double slashes in the text of the path:

```
C:\\Documents\\Data\\Dataflow
```

3.2   GIS Software

Some portions of Plotting interpolated surfaces (Section 4.5) requires dedicated GIS software such as QGIS, ArcGIS, or GRASS GIS. In the interest of creating a fully

reproducible and streamlined workflow for plotting Dataflow output some functions (`grassmap`) requires GRASS GIS (GRASS). GRASS can be installed by navigating to http://grass.osgeo.org.

> ☞ The `grassmap` function assumes that the user is on a Linux machine and that GRASS is installed under `/usr/lib/grass70`. For more details see Bivand (2015).

4 Handling streaming data

4.1 Cleaning incoming streaming data files

Incoming streaming data should be placed in your working directory under `DF_FullDataSets/Raw/InstrumentOutput` in a folder named for the survey date in yyyymm format (e.g. Feb-2015 is 201502). Likewise, data files (DF, C6, Eureka/Manta, Exo) in `".csv"` format should by preappended with the survey date in yyyymmdd format. Also, data files should be appended with the two letter code corresponding to its respective instrument (see below). For example, the raw data files for February 2015 would be named as follows and placed in the following directory stucture:

```
/
└── Dataflow
    └── DF_FullDataSets
        └── Raw
            └── InstrumentOutput
                └── 201502
                    ├── 20150211_DF021115_DF.csv
                    ├── 20150211_C6_11FEB_C6.csv
                    ├── 20150211_11FEB_eu.csv
                    ├── 20150211_11FEB_exo.csv
                    └── ...
```

> ☞ When working in a non-Windows environment, `.csv` files created under Windows may need to be opened and resaved as `text\csv` in order to avoid illegal "nul" characters.

Cleaning is accomplished via the `streamclean` function. The example below shows how to specify inputs to clean the data for the June 2016 survey. First, assign the `yearmon` argument to `201606`. Next, designate which intrument will be the primary source for gps measurements. In this case, we need to set `gps` to *eu*. Specify the

expected number of measurements per minute for each intrument using the respective `mmin` argument. For this example we set the `tofile` parameter to `FALSE` but setting it to `TRUE` will save the cleaned output to `DF_FullDataSets` as a `.csv` file.

The `streamclean` function will gather all the records associated with the remaining streams, merge them with the `gps` target, remove leading and trailing records of all zeros, format GPS coordinates, check that conductivity to salinity calculations are correct (recalculate if neccesary), and classify records based on fathom and CERP basin designations. Variable names and column ordering are formatted consistently and a machine readable (POSIX) date-time stamp is created.

```
> dt <- streamclean(yearmon = 201606, gps = "eu", eummin = 12, c6mmin = 12,
+        tofile = FALSE)
```

Some older Dataflow surveys have undergone a "hand-cleaning" and are missing the raw inputs neccessary to run `streamclean`. In these instances, the `streamparse` function can be used to align the formatting of these files to resemble the output of `streamclean`. Specifically, the function creates a POSIX compliant date field, and reproduces the column names and ordering of a `streamclean` output.

```
> dt <- streamparse(yearmon = 201007, tofile = FALSE)
```

4.2 QA cleaned streaming data

The `streamclean` function does not perform detailed QA of individual parameter values or detect systematic bias. Systematic bias might occur because of sensor drift or failure. In these cases there may be data but it is "junk data". The `streamqa` function creates a series of diagnostic plots of the data. The user enters an interactive QA "session" to detect systematic biases and eliminate unrealsitic data spikes. The set of dialogs steps through each variable named in the `parset` argument and gives the user the opportunity to define a valid data range (flagging values outside this range). Define a range higher (or lower) than all values to flag all records for that particular variable.

Cleaned data files are not modified directly. Instead, `streamqa` creates a matrix of the same size as the original data file and populates this matrix with flags. `streamqa` writes the "qafile" output to the `DF_FullDataSets/QA` directory and names it as the date appended by "qa". Subsequent analyses can filter the full dataset based on these QA flags.

```
> streamqa(yearmon = 201606, parset = names(streamget(201606))[c(4:12, 16:22)])
```

`streamqa` can be run more than once. On the first run, a new "qafile" will be produced. Subsequent runs will pull the data (filtered by the previous qa) and edit the existing qafile.

4.3 Loading previously cleaned streaming data

The `streamget` function will retrieve previously cleaned data. The function looks for full data sets in the `DF_FullDataSets` folder that match the specified `yearmon` survey date. The optional parameter `qa` is set to TRUE by default in order for `streamget` to filter the dataset by corresponding `streamqa` output. An example for the February 2015 survey is shown below.

```
> dt <- streamget(yearmon = 201606, qa = TRUE)
```

4.4 Interpolating cleaned data files

The `streaminterp` function will interpolate a dataset that has been loaded into memory from the `streamclean` or `streamget` functions. Interpolations are performed using functions in the `ipdw R` package (Stachelek 2014). Variables to be interpolated must be specified as inputs to the `paramlist` paramter. If you have loaded your dataset to memory under the name `dt`, use `names(dt)` to see the available parameters. Enter one or more parameters as arguments to a character vector. For example, to interpolate salinity only (as below) use `c("salinity.pss")`. Additional parameters can be appended. For example, to interpolate salinity and temperature use `c("salinity.pss","temp.deg.c")`. Interpolation should take about 20 minutes plus about 2 minutes for each entry in `paramlist`. Raster surface output will be written to a subfolder of `DF_Surfaces` named for the appropriate year and month of the survey.

`streaminterp` will first attempt to split the full data set into training and validation datasets. If these already exist in the `DF_Subsets` and `DF_Validation` folders, a warning will be printed and the pre-existing datsets will be used. Next, `streaminterp` will attempt to create a dedicated folder under `DF_Surfaces` to hold all the interpolated surfaces for the given survey. If this folder already exists, `streaminterp` will print a warning but the function should proceed as normal (the warning can be disregarded).

More details regarding the interpolation procedure can be found in Stachelek and Madden (2015).

```
> streaminterp(streamget(yearmon = 201606, qa = TRUE),
+   paramlist = c("salinity.pss"), 201606)
```

4.5 Plotting interpolated surfaces

4.5.1 Quick plot with R graphics

A quick visual inspection of interpolated outputs can be accomplished using the `surfplot` function. The `rnge` parameter takes either a single survey date or a list

of two survey dates to specify a date range for plotting. More detailed publication quality maps should be produced using a dedicated GIS program such as ArcGIS, QGIS, or GRASS GIS.

```
> surfplot(rnge = c(201502), params = c("sal"))
```

4.5.2 Detailed plotting with GRASS GIS

The `grassmap` function creates detailed publication quality maps using GRASS GIS. Individual map components (panels, legends, etc) are output to the `QGIS_plotting` folder. Final map outputs are written to the working directory. The following command creates a Bay-wide salinity map for May 2015.

```
> grassmap(rnge = 201505, params = c("sal"))
```

4.5.3 Producing timeseries for specific basins

The `basin` parameter of the `grassmap` function allows the user to limit (zoom-in) to a specific FATHOM basin. A listing of FATHOM basins can be found by inspecting `DF_Basefile/fathom_basins_proj.shp` or by referencing Cosby et al. (2005). The following command will create a series of zoomed-in salinity maps of Manatee Bay for each survey date between June 2006 and May 2015.

```
> grassmap(rnge = c(201205, 201305), params = c("sal"),
+          basin = "Manatee Bay", numcol = 3, numrow = 3)
```

5 Handling discrete grab sample data

5.1 Cleaning grab sample records

Incoming grab sample `.csv` data files should be placed in the `DF_GrabSamples/Raw` folder and their file names should have the survey date in yyyymm format preappended. These files can be cleaned using the `grabclean` function. The `grabclean` function formats column names, removes columns/rows of missing data, and calculates minute averages of the streaming data that correspond to the grab sample date/times. Output is saved to the `DF_GrabSamples` folder when `tofile` is set to `TRUE`.

```
> grabclean(yearmon = 201410, tofile = FALSE)
```

Suspect data records should be identified manually in the `flags` column. This becomes important in Section 6.2 because suspect data records can create problems converting between extracted and fluoresced chlorophyll.

5.2 Loading previously cleaned grab data

The `rnge` paramter takes either a single survey date or a list of two survey dates to specify a date range for retrieving cleaned grab data.

```
> grabs <- grabget(rnge = c(201402, 201410))
```

6 Data Analysis

6.1 Calculating a difference-from-average surface

The `avmap` function takes a survey date as input and searches the `DF_Surfaces` folder for interpolated surfaces of the same parameter within a specified number of months for each year. The number of months on either side of the input month is set using the `tolerance` parameter. The found surfaces often have different extents. The `percentcov` parameter controls the percent of all identified surveys required before a pixel is included in difference-from-average computations. Output surfaces are written to the current working directory unless the `tofile` parameter is set to `FALSE`.

```
> avmap(yearmon = 201502, params = "sal", tofile = TRUE, percentcov = 0.6,
+       tolerance = 1)
```
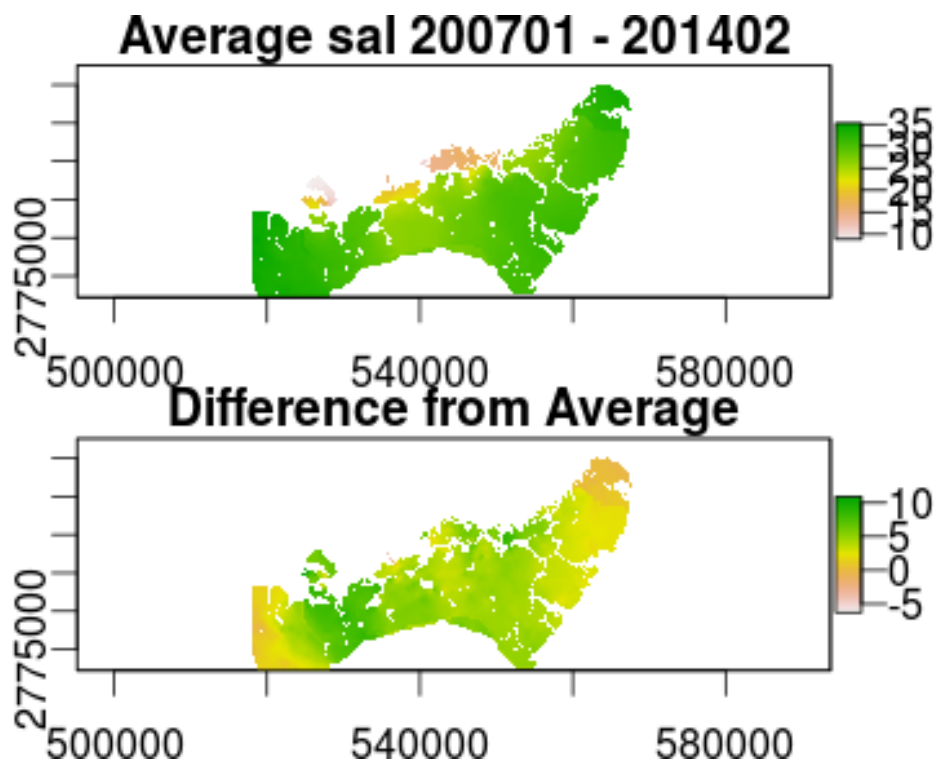
6.2 Fit grab sample and streaming averages

6.2.1 Calculate coefficients

In order to generate maps of chlorophyll concentration, streaming fluorescence values (chlorophyll, algal pigments, cdom) must be statistically "fit" (regressed) against lab-derived extracted chlorophyll values. The `chlcoef` function searches the `DF_GrabSamples` folder for a cleaned grab dataset that matches the specified `yearmon` parameter value. First, the function generates a correlation matrix and removes streaming variables that have a correlation coefficient with extracted chlorophyll less than the value passed to the `corcut` argument. The resulting variables are entered into a multiple linear regression. The resulting "full" model is subjected to a backward stepwise AIC model selection (Venables and Ripley 2002). The output of this step is usually a regression with a reduced number of parameters. The final regression is checked for multicolinearity by calculating variance inflation factors (vif) and excluding parameters until all vif values are less than 10 (Helsel and Hirsch 2002). The previous steps, which include summaries of all intermediate model fits, are printed to the R console for manual inspection.

The final set of variable coefficients, the R^2 value, the p-value, and the formula for the final fitted equation are printed (appended) to the `extractChlcoef.csv` file in the `DF_GrabSamples` folder.

```
> chlcoef(yearmon = 201502, remove.flags = TRUE)
```

6.2.2 Generate extracted chlorophyll surfaces

The coefficients calculated as a result of the `chlcoef` function can be used to create an interpolated map of chlorophyll concentration. The `chlmap` function takes these coefficients and the associated fulldataset and calculates an extracted chlorophyll value for each measurement point ("extchl"). These values are interpolated using the `streaminterp` function. The output surface is stored in the `DF_Surfaces` folder under the appropriate survey date folder.

```
> chlmap(yearmon = 201502)
```

References

Bivand, R. (2015). *rgrass7: Interface Between GRASS 7 Geographical Information System and R*. R package version 0.1-0.

Cosby, B., Nuttle, W., and Marshall, F. (2005). Fathom enhancements and implementation to support development of minimum flows and levels for florida bay. *Environmental Consulting and Technology, Inc., South Florida Water Management District. West Palm Beach, Florida.*

Helsel, D. and Hirsch, R. (2002). Statistical methods in water resources: US Geological Survey Techniques of Water Resources Investigations, book 4, chap.

Stachelek, J. (2014). *ipdw: spatial interpolation by Inverse Path Distance Weighting.* R package version 0.2-2.

Stachelek, J. and Madden, C. J. (2015). Application of inverse path distance weighting for high-density spatial mapping of coastal water quality patterns. *International Journal of Geographical Information Science*, pages 1–11.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S.* Springer, New York, fourth edition. ISBN 0-387-95457-0.