Deliverables and Guidelines for Future Use

Scott McCain

May 26, 2017

Index

Summary of Deliverables

- Wrote a flexible series of R scripts to perform various objectives: file conversion, file transfer, and plotting. This framework (detailed below) can be used for any other fixed station websites.
- Updated the Bedford Basin Monitoring Program, by completing plots from 1999-2009, and ODFs from 1999-2009.
- Converted all current ODFs to CSVs onto the website, and created a continually updating aggregated CSV with consistent formatting.
- Worked with Patrick Upson to update the look and feel of the website.
- Contributed all code and development through the collaborative and version-controlled system Git and GitHub.
- Met with stakeholders to establish goals and use-cases for website
    - Catherine Johnson, Emmanuel Devred, Dave Brickman, Dave Herbert, Andrew Cogswell (BIO)
    - Dave Covey and Richard Davis (Dalhousie, MEOPAR)

Framework for Fixed Station Websites

1) Populating an FTP:
    - Declare working directories, determine file naming structure.
    - Create folders in FTP with similar structure to BBMP FTP.
    - Using the transfer, conversion, and plotting functions, fill in the folders.
    - Using the bulk data aggregation script, create aggregated csv of station.
2) Create a site-context map:
    - Adjust the latitude and longitude limits for the map R script, and produce a corresponding site map (easily adapted from `Bedford_Basin_Map.R`)
3) Write detailed site description and sampling overview:
    - Similar to the current text on the BBMP website, a detailed overview of sampling should be written (and translated). This text should overview why this site was chosen, what sampling is done, and any other important site characteristics.

R Scripts for Fixed Station Website Populating

The following functions can be used to:

1) Populate a designated File Transfer Protocol (FTP) with backlogged profile images, ODF cast files, and CSVs, and corresponding summary (.tsv) files.
2) Create an aggregated .csv file of all casts, with date labels.

These functions are a general framework for doing the objectives above, however have been specifically applied to the Bedford Basin Monitoring Program. Therefore, the default arguments are for the BBMP. Below, I describe the functions, their use, and how to adopt them for future use.

General Use

- `odf_file_lister()`: Produces a set of odf files that are matched for certain naming conventions. e.g. searches for only Bedford Basin 667.

- `directory_lister_wrapper()`: A wrapper function for the `odf_file_lister()`, which switches to the Src if files aren't present.

Plotting Functions

- `fluorescence_oxygen_plot()`: Creates a the last fluorescence/oxygen plot in the four-panels of plots. Function needed for years that have one or both variables missing.

- `odf_plot_function()`: Creates the four panel plot.

- `odf_year_plots()`: Creates the four panel plot for an year.

File Transfer and Conversion

- `odf_date_finder()`: Simple function that returns the date of an ODF.
- `odf_file_renamer()`: Reads in an ODF file, and depending if it was read from the Arc (double quality controlled), or from the Src, the name will be reformated.

- `transfer_files_odf()`: Files are renamed and copied from the Arc or Src and transferred into the BBMP site FTP.

- `transfer_files_csv()`: Files are renamed and converted from the Arc or Src and transferred into the BBMP site FTP.

Function Descriptions and Key Considerations for Future Use

There are two expected problems with adapting this code for another fixed station.

1) Working directory switching.
2) File naming and searching.

`odf_file_lister` lists all files in a working directory from some pattern (e.g. searching for the site code "667"). These patterns differ if the file was found in the Src or the Arc. This function would need to be altered to accomodate a new file naming convention.

```
odf_file_lister
```

```
function (working_directory, year_i = year, site_code = "667")
{
    expect_true(dir.exists(working_directory), "File folder does not exist in the FTP.")
    if (!use_src) {
        odf_file_list_i <- list.files(pattern = "*^.*D.*.ODF$")
        if (length(odf_file_list_i) == 0) {
            odf_file_list_i <- list.files(pattern = ".ODF$")
        }
        only_667 <- grepl(pattern = paste(site_code, "_", sep = ""),
            x = odf_file_list_i)
        only_DN <- grepl(pattern = "_DN", x = odf_file_list_i)
        if (year_i > 1999) {
            only_bcd <- grepl(pattern = "BCD", x = odf_file_list_i)
            odf_file_list_i <- odf_file_list_i[only_667 & only_bcd &
                only_DN]
        }
        else if (year_i == 1999) {
            only_99667 <- grepl(pattern = "99667", x = odf_file_list_i)
            odf_file_list_i <- odf_file_list_i[only_99667]
        }
    }
    else if (use_src) {
        odf_file_list_i <- list.files(pattern = "*^.*D.*.ODF$")
    }
    return(odf_file_list_i)
}
```

`directory_lister_wrapper` is the function that would enable new directory switching. The arguments `arc_root` and `src_root` are critical for this. An important aspect of this function is that global variables are declared depending on where the files are located (via `use_src <<- TRUE/FALSE`). This allows the functions to "talk" to each other, which is particularly important for naming.

If the naming conventions are the same as the BBMP, then a first try would simply by to switch the directory locations in this function (the default is set for the BBMP).

```
directory_lister_wrapper
```

```
function (year_x = year, site_code_i = "667", arc_root = "R:\\Science\\BIODataSvc\\ARC\\Archive\\ctd\\"
    src_root = "R:\\Science\\BIODataSvc\\SRC\\BBMP\\COMPASS\\")
{
    arc_wd <- paste(arc_root, year_x, sep = "")
    src_wd <- paste(src_root, year_x, sep = "")
    if (dir.exists(arc_wd)) {
        setwd(arc_wd)
        use_src <<- FALSE
        used_directory <<- arc_wd
        odf_files <- odf_file_lister(working_directory = arc_wd,
            year_i = year_x, site_code = site_code_i)
        no_odf_files <- length(odf_files)
    }
    if (!dir.exists(arc_wd)) {
        setwd(src_wd)
        use_src <<- TRUE
        used_directory <<- src_wd
        odf_files <- odf_file_lister(working_directory = src_wd,
            year_i = year_x)
    }
    return(odf_files)
}
```

`odf_file_renamer` renames files dynamically dependent on the original file location (src vs. arc).

```
odf_file_renamer
```

```
function (odf_file_i, file_extension = "ODF", src_format = TRUE)
{
    expect_that(odf_file_i, is_a("character"))
    if (file_extension %!in% c("ODF", "csv")) {
        stop("File extension is not available.")
    }
    if (src_format) {
        odf_read_in <- read.odf(odf_file_i)
        year_date <- format(odf_read_in[["date"]], "%Y")
        front_string <- "CTD_BCD"
        cast_number <- substr(odf_file_i, nchar(odf_file_i) -
            6, nchar(odf_file_i) - 4)
        final_file_name <- paste(front_string, year_date, "667_",
            cast_number, "_01_DN.", file_extension, sep = "")
    }
    else {
        final_file_name <- paste(str_sub(odf_file_i, start = 1,
            end = -4), file_extension, sep = "")
    }
    return(final_file_name)
}
```

`transfer_files_csv` and `transfer_files_odf` are very similar functions, in that they both transfer files. The key difference is that ODF files can simply be copied, while csv files have to be opened and rewritten into a different directory.

```
transfer_files_csv
```

```
function (year, out_root = "R:\\Shared\\Cogswell\\_BIOWeb\\BBMP\\CSV\\")
{
    odf_files <- directory_lister_wrapper(year_x = year)
    no_odf_files <- length(odf_files)
    out_file_dir <- paste(out_root, year, "\\", sep = "")
    expect_true(no_odf_files > 0, info = "No files found.")
    odf_date_summary <- vector(length = no_odf_files)
    odf_name_summary <- vector(length = no_odf_files)
    for (i in 1:no_odf_files) {
        setwd(used_directory)
        new_odf_file_name <- odf_file_renamer(odf_file_i = odf_files[i],
            src_format = use_src, file_extension = "csv")
        opened_ctd_odf <- read.ctd.odf(odf_files[i])
        setwd(out_file_dir)
        write.ctd(opened_ctd_odf, file = paste(out_file_dir,
            new_odf_file_name, sep = ""))
        print(new_odf_file_name)
        odf_date_summary[i] <- summary_date
        odf_name_summary[i] <- new_file_name
    }
    odf_summary_dates_names <- data.frame(FILE = odf_name_summary,
        START_DATE_TIME = odf_date_summary)
    odf_summary_file <- paste(out_root, year, "\\", year, "667CSVSUMMARY.tsv",
        sep = "")
    cat(paste("Folder consists of ", no_odf_files, " CSV (converted from ODF) files from ",
        year, " Bedford Basin Compass Station occupations.",
        sep = ""), file = odf_summary_file, sep = "\n", append = FALSE)
    cat("", file = odf_summary_file, sep = "\n", append = TRUE)
    write.table(odf_summary_dates_names, file = odf_summary_file,
        append = TRUE, quote = TRUE, sep = ",", eol = "\n", na = "NA",
        dec = ".", row.names = FALSE, col.names = TRUE)
}
```

```
transfer_files_odf
```

```
function (year, out_root = "R:\\Shared\\Cogswell\\_BIOWeb\\BBMP\\ODF\\")
{
    odf_files <- directory_lister_wrapper(year_x = year)
    no_odf_files <- length(odf_files)
    out_file_dir_base <- paste(out_root, year, "\\", sep = "")
    odf_date_summary <- vector(length = no_odf_files)
    odf_name_summary <- vector(length = no_odf_files)
    for (i in 1:no_odf_files) {
        new_file_name <- odf_file_renamer(odf_file_i = odf_files[i],
            src_format = use_src, file_extension = "ODF")
        summary_date <- odf_date_finder(odf_file_i = odf_files[i])
        start_file <- paste(used_directory, "\\", odf_files[i],
            sep = "")
        out_file <- paste(out_file_dir_base, new_file_name, sep = "")
        file.copy(from = start_file, to = out_file)
        print(out_file)
        odf_date_summary[i] <- summary_date
        odf_name_summary[i] <- new_file_name
    }
    odf_summary_dates_names <- data.frame(FILE = odf_name_summary,
        START_DATE_TIME = odf_date_summary)
    odf_summary_file <- paste(out_root, year, "\\", year, "667ODFSUMMARY.tsv",
        sep = "")
    cat(paste("Folder consists of ", no_odf_files, " ODF files from ",
        year, " Bedford Basin Compass Station occupations.",
        sep = ""), file = odf_summary_file, sep = "\n", append = FALSE)
    cat("", file = odf_summary_file, sep = "\n", append = TRUE)
    write.table(odf_summary_dates_names, file = odf_summary_file,
        append = TRUE, quote = TRUE, sep = ",", eol = "\n", na = "NA",
        dec = ".", row.names = FALSE, col.names = TRUE)
}
```