waterquality_Demo_WQW-23

Richard Johansen

2023-02-13

Introduction to WQ

The main purpose of **waterquality** is to quickly and easily convert satellite-based reflectance imagery into one or many water quality indices designed for the detection of harmful algal blooms (HABs).

Currently, this package is able to process 40 algorithms for the following satellite-based imagers: WorldView-2 and -3, Sentinel-2, Landsat-8, MODIS, MERIS, and OLCI.

The main function of this packge is `wq_calc`, which converts satellite imagery into a HAB index using band ratio algorithms. Additionally, **waterquality** has a series of intuitive mapping (`Map_WQ`) and modeling (`extract_lm`) functions to assist users with the aesthetically pleasing maps and standardized statistical outputs.

NOTE Modeling functions require ground-truth data in order to validate imagery.

Technical Documents and Additional Resources

"Waterquality: An Open-Source R Package for the Detection and Quantification of Cyanobacterial Harmful Algal Blooms and Water Quality"
GitHub Repository
Waterquality Vignette

Getting Started with waterquality

Install & Load Required R Packages

```
#Install and load the waterquality and raster packages
require(waterquality)
require(raster)
require(tidyverse)
require(tmap)
require(tmaptools)
require(sf)
require(caret)
```

Import Satellite Image

Although these details are beyond the scope of this talk, it is critical to make sure the imagery is downloaded and processed according to the technical report.

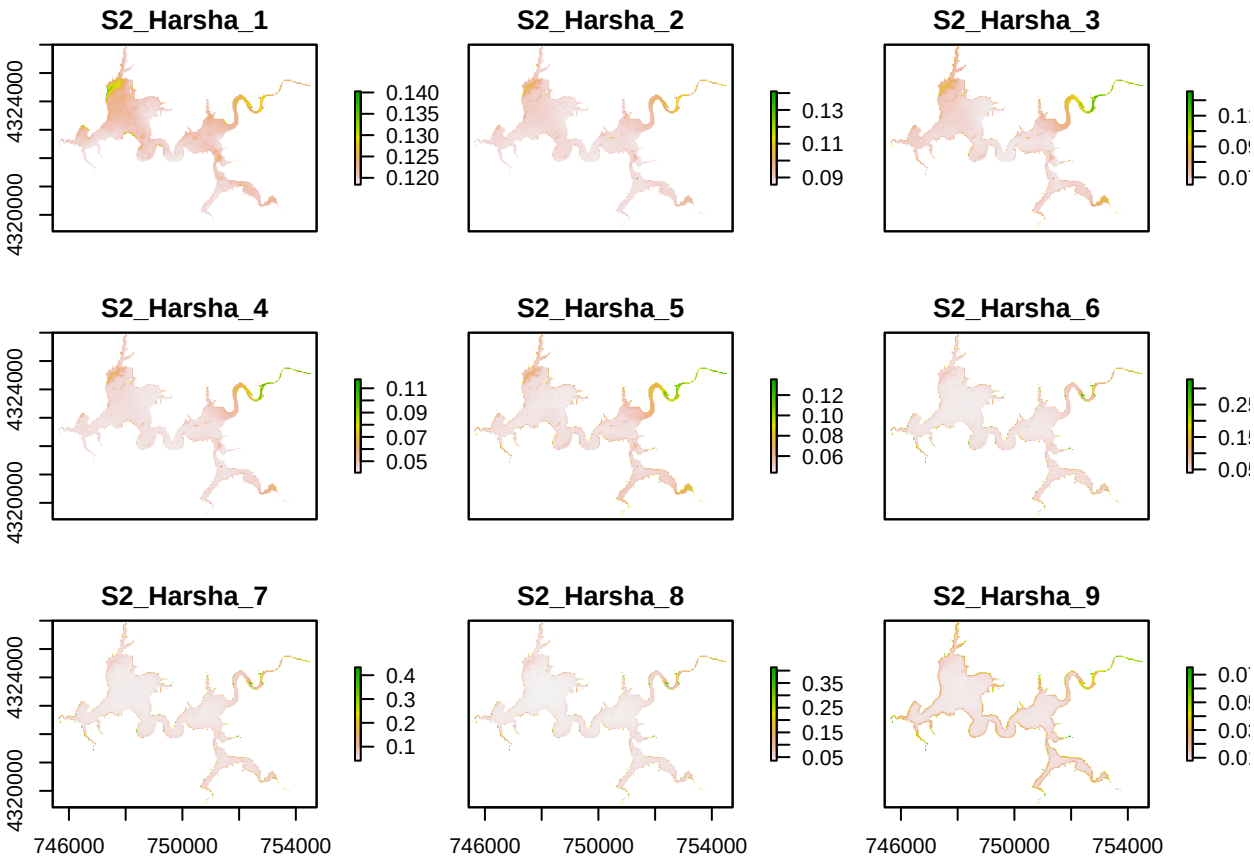
Here are few common locations:

```
#USGS Earth Explorer
browseURL("https://earthexplorer.usgs.gov/")

#ESA's Copernicus HUB
browseURL("https://scihub.copernicus.eu/dhus/#/home")

# Sen2r R Package (Recommended)
# Provides Sen2Cor Atmospheric Correction
# Allows Spatial masking
browseURL("https://sen2r.ranghetti.info/")
```

waterquality contains sample sentinel-2 imagery for Harsha lake in SW Ohio. A land mask has already been applied to the imagery and it only includes the first 9 spectral bands.

```
# Load Sentinel-2 Imagery of Harsha Lake in SW Ohio
Harsha <- stack(system.file("raster/S2_Harsha.tif", package = "waterquality"))
# Plot RGB Image
plotRGB(Harsha,r=4,g=3, b=2,stretch='lin')
```

```
# Plot each band
plot(Harsha/10000) #reflectance is DN/10000
```




```
#plot(Harsha)
```

```
browseURL('https://gisgeography.com/sentinel-2-bands-combinations/')
```

Sentinel-2 Bands

wq_calc()

The main function of this package is called `wq_calc()` which calculates water quality indices by using a reflectance raster stack as an input, user-defined algorithm(s) selection, and satellite configuration selection corresponding to the following three arguments: `raster_stack`, `alg`, and `sat`.

- `raster_stack` - The input reflectance image to be used in band algorithm calculation.
- `alg` - Determines the indices to be utilized:
  - Single Algorithm
  - Multiple Algorithm
  - Type of Algorithm
  - All Possible Algorithms
- `sat` - Determines the appropriate spectral configuration and subsequently appropriate algorithms to be calculated from predefined list:

- WorldView-2
- Sentinel-2
- Landsat-8
- MODIS
- MERIS
- OLCI

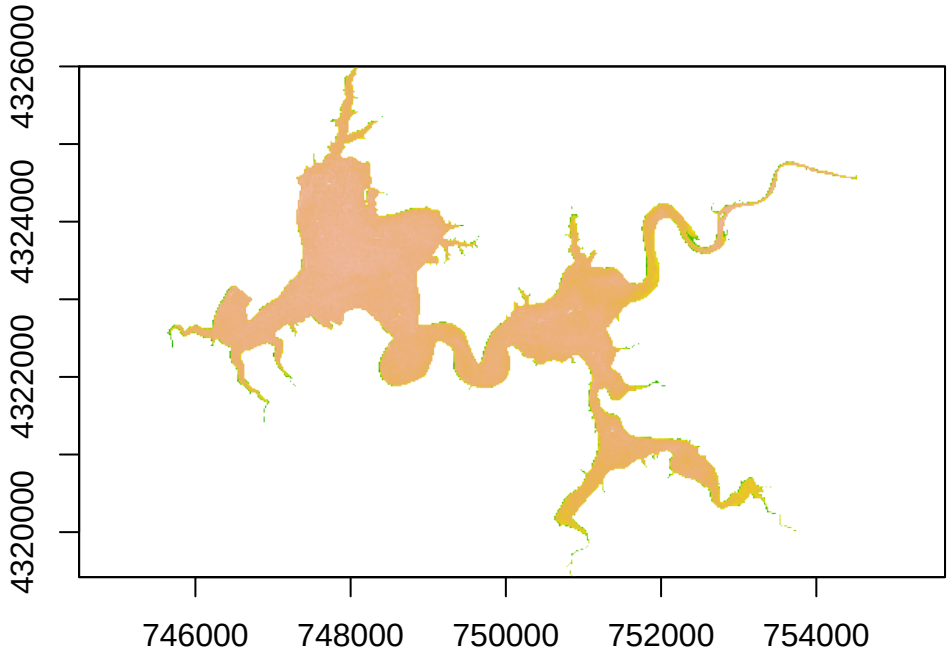
```
#Can examine functions with a '?'
?wq_calc
```

Explore wq_calc

```
Harsha_NDCI <- wq_calc(raster_stack = Harsha,
                           alg = "MM12NDCI",
                           sat = "sentinel2")
```

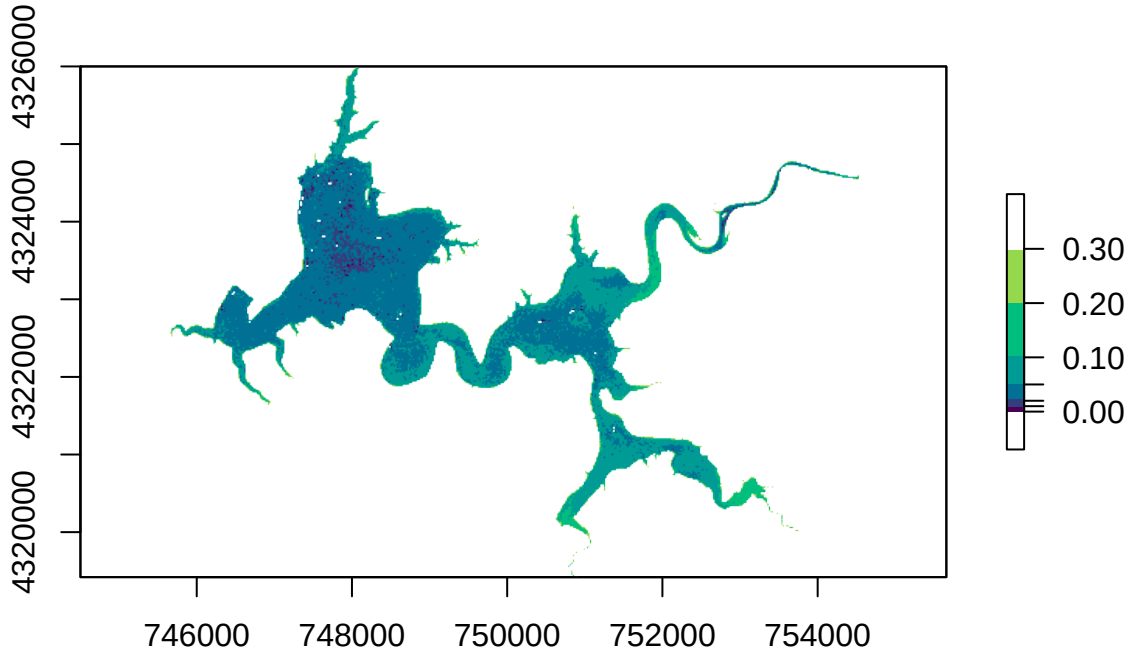
Algorithm

```
# Plot NDCI Image
plot(Harsha_NDCI)
```

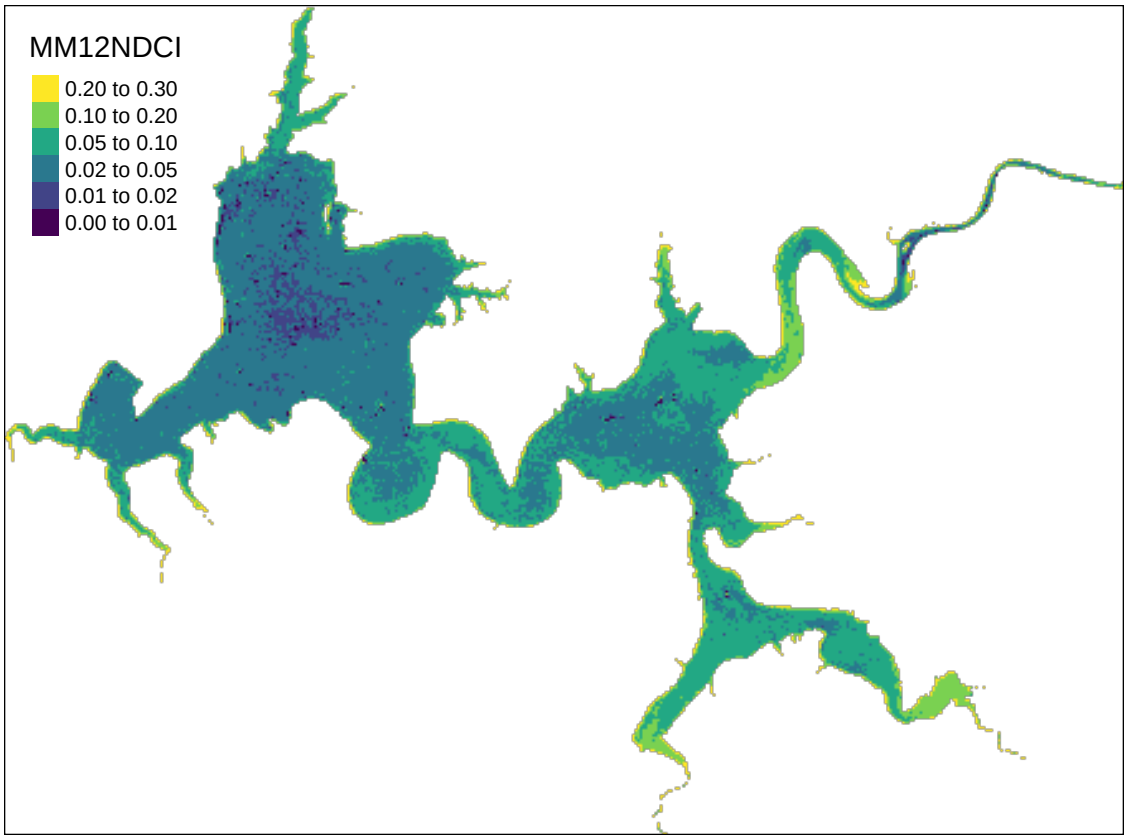


Visualize NDCI Algorithm

```
# Plot NDCI image with
b <- c(0,0.01,0.02,0.05,0.1,0.2,0.3)
plot(Harsha_NDCI,
     col=hcl.colors(n=length(b), palette = "viridis"),
     breaks = b)
```




```
#### Highly recommend using TMAP Package for mapping/geospatial analyses in R!
browseURL('https://r-tmap.github.io/tmap-book/index.html')
tm_shape(Harsha_NDCI) +
  tm_raster(palette = "viridis", breaks = b, legend.reverse = TRUE)
```



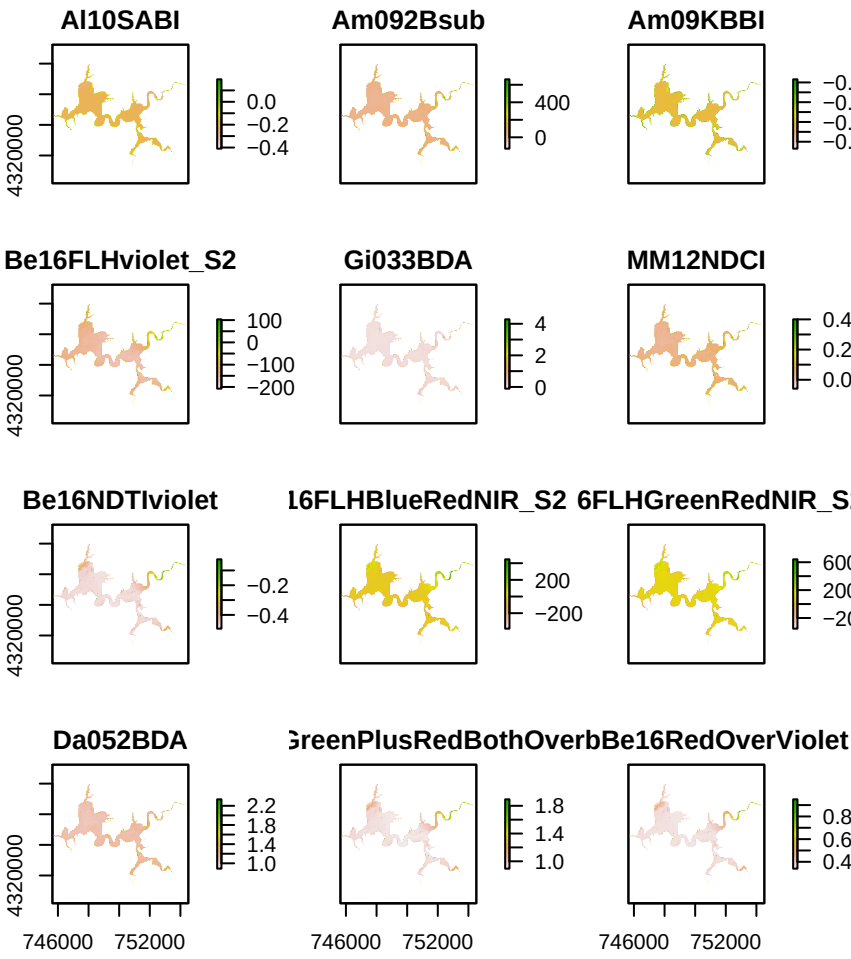

```
Harsha_Multi <- wq_calc(raster_stack = Harsha,
                        alg = c("Am092Bsub", "MM12NDCI", "Da052BDA"),
                        sat = "sentinel2")
```

Multiple Algorithms

```
Harsha_all <- wq_calc(raster_stack = Harsha,
                      alg = 'all',
                      sat = "sentinel2")

Harsha_Chl <- wq_calc(Harsha,
                    alg = "chlorophyll",
                    sat = "sentinel2")
```

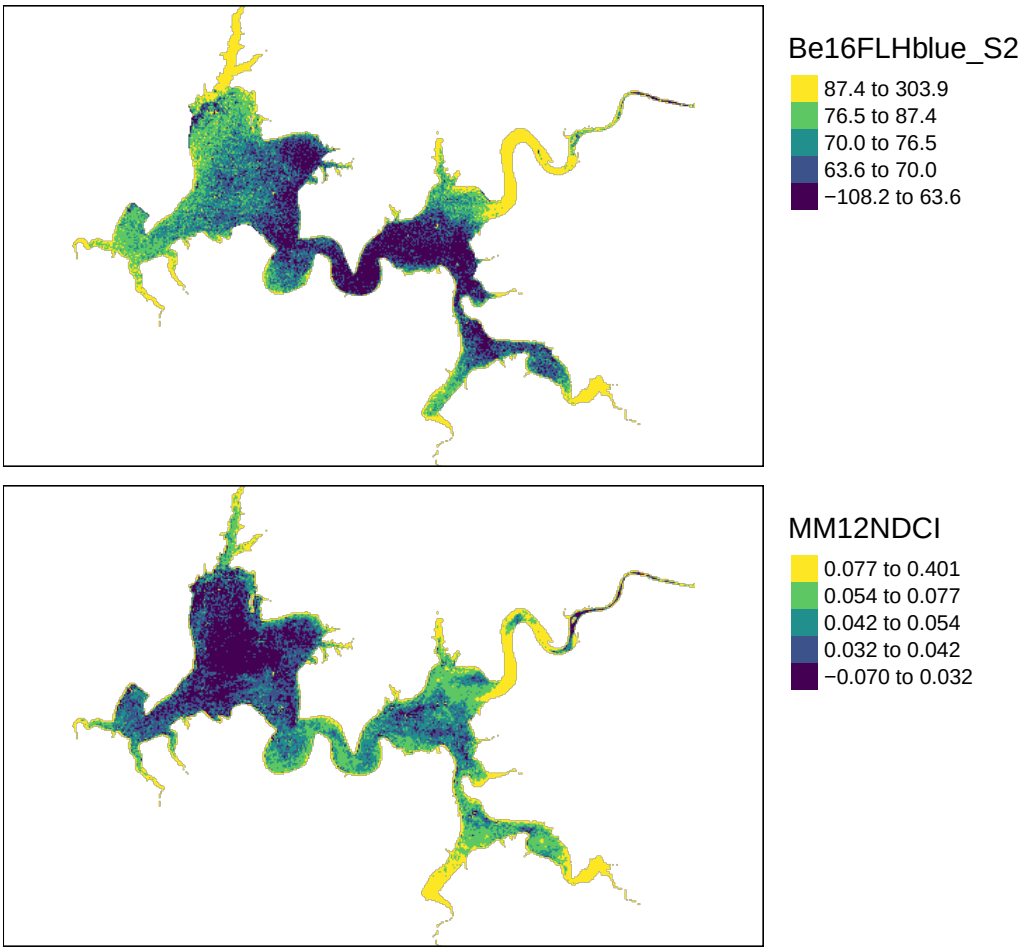
Algorithms by Type or All

```
plot(Harsha_all)
```



Plotting Multiple Algorithms using TMAP

```
map1 <- tm_shape(Harsha_all[[4]]) +
  tm_raster(palette = "viridis", style = "quantile", legend.reverse = TRUE, drop.levels = TRUE)+
  tm_layout(title.size = 0.6,legend.outside = TRUE)

map2 <- tm_shape(Harsha_all[[7]]) +
  tm_raster(palette = "viridis", style = "quantile", legend.reverse = TRUE, drop.levels = TRUE)+
     tm_layout(title.size = 0.6,legend.outside = TRUE)

tmap_arrange(map1,map2)
```



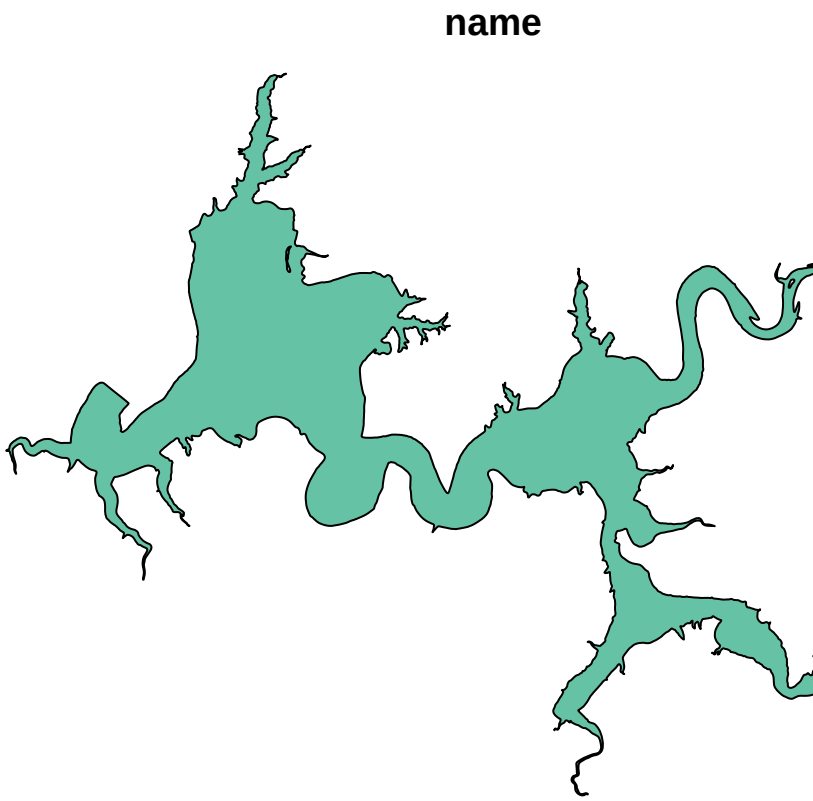
waterquality Mapping Functions

Map_WQ_raster

This function wraps the "tmap" package to help users to efficiently generate a map of a raster image which can be overlaid with optional geospatial objects and data histogram. In order to simplify this process and reduce the technical expertise required, the number of arguments were reduced to the following: - `WQ_raster` - Raster file generated from `wq_calc` or other GeoTiff file
- `sample_points` - geospatial file (.shp or .gpkg) containing sampling locations - `map_title` - text used to generate title of map - `raster_style` - method to process the color scale when col is a numeric variable. Please refer to the style argument in the `?tmap::tm_raster()` function for more details (Default is "quantile"). - `histogram` - Option to add or remove a histogram of the data values. (Default is TRUE)

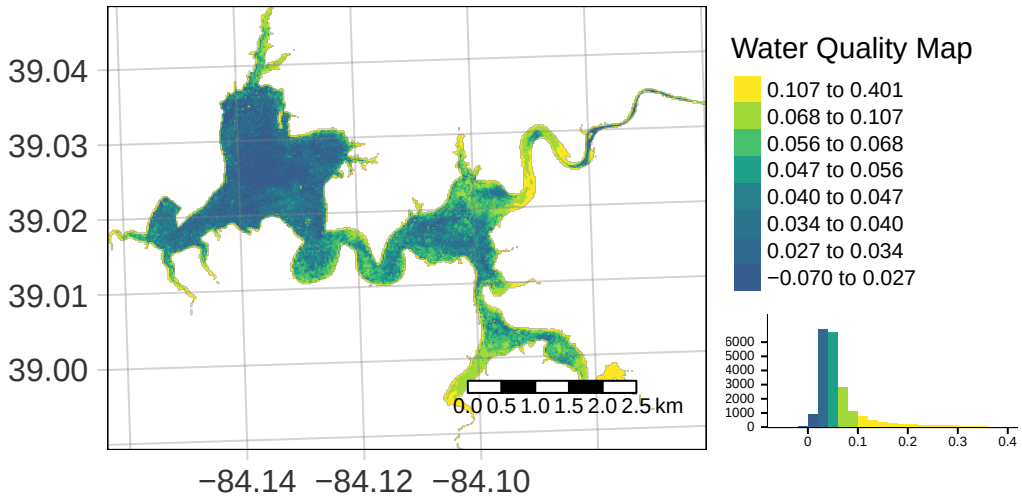
```
s2 = stack(system.file("raster/S2_Harsha.tif", package = "waterquality"))
MM12NDCI = wq_calc(s2, alg = "MM12NDCI", sat = "sentinel2")
samples = st_read(system.file("raster/Harsha_Simple_Points_CRS.gpkg", package = "waterquality"))
# Explore In Situ Samples
View(samples)
# View Harsha Lake Extent
```

```
lake_extent = st_read(system.file("raster/Harsha_Lake_CRS.gpkg", package = "waterquality"))
plot(lake_extent) # Just a boundary of the lake
```



Sample Mapping using default Data

```
Map_WQ_raster(WQ_raster = MM12NDCI,
              sample_points = samples,
              map_title= "Water Quality Map",
              raster_style = "quantile",
              histogram = TRUE)
```




```
#install.packages('OpenStreetMaps')
Map_WQ_basemap(WQ_extent = lake_extent,
               sample_points = samples,
               WQ_parameter = "Chl_ugL",
               map_title= "Water Quality Map",
               points_style = "quantile",
               histogram = TRUE)
```

Basemap with points

waterquality Statical Functions

These functions have been developed to easily and quickly evaluate algorithm performace. Additionally, these function result in standardized outputs with the following: **Global Model**
- r^2
- p-value
- slope
- intercept

Crossvalidated Model
- average r^2
- average RMSE
- average MAE

Sample Code for Staging your data MAKE SURE YOU DATA IS PROJECTED CORRECTLY!

```
#Input raster image
wq_raster <- stack("C:/temp/my_raster.tif")
#Input shapefile
wq_samples <- shapefile('C:/temp/my_samples.shp')
#Extract values from raster and combine with shapefile
waterquality_data <- data.frame(wq_samples, extract(wq_raster, wq_samples))
#Export results as csv file
write.csv(waterquality_data, file = "C:/temp/waterquality_data.csv")
```

```
df <- read.csv(system.file("raster/waterquality_data.csv", package = "waterquality"))
View(df)
```

Load Sample Data

Simple linear regression analysis (One Algorithm & One WQ Parameter) extract_lm - `parameter` A string specifying water quality parameter - `algorithm` A string specifying water quality algorithm - `df` data frame containing the values for parameter and algorithm arguments

```
extract_lm(parameter = "Chl_ugL", algorithm = "MM12NDCI", df = df)
```

Robust linear regression analysis (Cross-validated One Algorithm & One WQ Parameter)

extract_lm_cv - `parameter` A string specifying water quality parameter - `algorithm` A string specifying water quality algorithm - `df` data frame containing the values for parameter and algorithm arguments - `train_method` A string specifying which classification or regression model to use (Default = "lm"). See ?caret::train for more details - `control_method` A string specifying the resampling method (Default = "repeatedcv"). See ?caret::trainControl for more details - `folds` the number of folds to be used in the cross validation model (Default = 3) - `nrepeats` the number of iterations to be used in the cross validation model (Default = 5)

```
extract_lm_cv(parameter = "Chl_ugL", algorithm = "MM12NDCI",
                                        df = df, train_method = "lm", control_method = "repeatedcv",
                                        folds = 3, nrepeats = 5)
```

Example

```
## # A tibble: 1 x 7
##   R_Squared Slope Intercept P_Value CV_R_Squared  RMSE   MAE
##       <dbl> <dbl>     <dbl>   <dbl>        <dbl> <dbl> <dbl>
## 1     0.162  26.2      4.69   0.153        0.374  1.87  1.31
```

Robust linear regression analysis on Full Data (Cross-validated user-defined Algorithms & user-defined WQ Parameter)

extract_lm_cv_multi

- `parameters` list of water quality parameters
- `algorithms` list of water quality algorithms
- `df` data frame containing the values for parameter and algorithm arguments
- `train_method` A string specifying which classification or regression model to use (Default = "lm"). See ?caret::train for more details
- `control_method` A string specifying the resampling method (Default = "repeatedcv"). See ?caret::trainControl for more details
- `folds` the number of folds to be used in the cross validation model (Default = 3)
- `nrepeats` the number of iterations to be used in the cross validation model (Default = 5)

```
# Create series of strings to be used for parameters and algorithms arguments
extract_lm_cv_multi2 <- function(parameters, algorithms, df, train_method = "lm", control_method = "repe
  if (!requireNamespace("caret", quietly = TRUE))
    stop("package caret required, please install it first")
  list = list()
  for (i in seq_along(parameters)) {
    names(algorithms) <- algorithms %>%
      purrr::map_chr(., ~ paste0(parameters[[i]], "_",.))
    list[[i]] = algorithms %>%
      purrr::map_dfr(~extract_lm_cv(parameter = parameters[[i]], algorithm = ., df = df,
                                    train_method = train_method, control_method = control_method,
                                    folds = folds, nrepeats = nrepeats), .id = "Algorithms")
  }
  extract_lm_cv_multi_results <- (do.call(rbind, list))
}

algorithms <- c(names(df[6:10]))
algorithms
```

```
## [1] "Al10SABI"  "Go04MCI"   "Ku15PhyCI" "MM12NDCI"  "Wy08CI"
```

```
parameters <- c(names(df[3:5]))
parameters
```

```
## [1] "Turbid_NTU" "Chl_ugL"    "BGA_PC"
```

```
extract_lm_cv_multi_results <- extract_lm_cv_multi2(parameters = parameters, algorithms = algorithms,
                                                    df = df, train_method = "lm", control_method = "repea
                                                    folds = 3, nrepeats = 5)
head(extract_lm_cv_multi_results)
```

```
## # A tibble: 6 x 8
##   Algorithms           R_Squared    Slope Intercept  P_Value CV_R_~1  RMSE   MAE
##   <chr>                    <dbl>    <dbl>     <dbl>    <dbl>   <dbl> <dbl> <dbl>
## 1 Turbid_NTU_Al10SABI      0.441 -30.5      -4.37    9.54e-3   0.463  2.71 1.81
## 2 Turbid_NTU_Go04MCI       0.763   0.0430    0.311   4.46e-5   0.450  2.19 1.43
```

```
## 3 Turbid_NTU_Ku15PhyCI     0.432   0.0818   -0.0687  1.06e-2   0.434  2.25 1.42
## 4 Turbid_NTU_MM12NDCI      0.166  42.3       0.370   1.48e-1   0.269  2.49 1.57
## 5 Turbid_NTU_Wy08CI        0.432   0.0818   -0.0687  1.06e-2   0.269  2.38 1.49
## 6 Chl_ugL_Al10SABI         0.534 -21.0       1.30    2.99e-3   0.559  1.29 0.985
## # ... with abbreviated variable name 1: CV_R_Squared
```

Robust linear regression analysis on Full Data (Cross-validated on all Algorithms Using defined WQ Parameter)

extract_lm_cv_all

- `parameters` list of water quality parameters
- `df` data frame containing the values for parameter and algorithm arguments
- `train_method` A string specifying which classification or regression model to use (Default = "lm"). See ?caret::train for more details
- `control_method` A string specifying the resampling method (Default = "repeatedcv"). See ?caret::trainControl for more details
- `folds` the number of folds to be used in the cross validation model (Default = 3)
- `nrepeats` the number of iterations to be used in the cross validation model (Default = 5)

```
extract_lm_cv_all_results <- extract_lm_cv_all(parameters = parameters, df = df,
                                               train_method = "lm", control_method = "repeatedcv",
                                               folds = 3, nrepeats = 5)
head(extract_lm_cv_all_results)
```

Example

```
## # A tibble: 6 x 8
##   Algorithms                 R_Squ~1  Slope Inter~2  P_Value CV_R_~3  RMSE   MAE
##   <chr>                        <dbl>  <dbl>   <dbl>    <dbl>   <dbl> <dbl> <dbl>
## 1 Turbid_NTU_pH                0.976 -14.1  125.    4.86e-11   0.630  1.39 0.798
## 2 Turbid_NTU_Al10SABI          0.441 -30.5   -4.37  9.54e- 3   0.556  3.19 2.16
## 3 Turbid_NTU_MM12NDCI          0.166  42.3    0.370 1.48e- 1   0.481  3.11 2.10
## 4 Turbid_NTU_Da052BDA          0.168  19.2  -18.7   1.45e- 1   0.330  2.68 1.64
## 5 Turbid_NTU_MM12NDCIalt       0.166  42.3    0.370 1.48e- 1   0.289  2.85 1.86
## 6 Turbid_NTU_TurbDox02NIRov~   0.136  -7.44   5.47  1.95e- 1   0.423  2.49 1.61
## # ... with abbreviated variable names 1: R_Squared, 2: Intercept,
## #   3: CV_R_Squared
```