rB3 demo for GLEON-20 pre-meeting workshop

Code

prepared for GLEON 20, GSA workshop 2018-12-03

Chris McBride & Kohji Muraoka, UoW, November 2018
correspondance to cmcbride@waikato.ac.nz (mailto:cmcbride@waikato.ac.nz)

INSTALLING rB3

Install remotes package to remotely install G20 version of rB3

** Web installation via git (‘remotes’ OR ‘devtools’) **

Hide

```
### via package "remotes"
# install.packages("remotes")
 remotes::install_github("kohjim/rB3", ref = "G20")

### via package "devtools"
# install.packages("devtools")
 devtools::install_github("kohjim/rB3", ref = "G20")

library(rB3)
```

** Local installation **

Hide

```
### download directly from github
# https://github.com/kohjim/rB3/archive/G20.zip

# !!! Open the folder 'Demo' and the file Demo.Rproj !!! #

# check your working directory, and adjust if needs be
getwd()
# setwd("C:/")

#install rB3
library(devtools)
install("../rB3-G20")

# load the rB3 library
library(rB3)
```

Hide

```
#install.packages("remotes")

##### If installing packages failed, manually install dependencies below:
# install.packages("tidyr")
# install.packages("ggplot2", dependencies = TRUE)
# install.packages("lubridate")
# install.packages("shiny")
# install.packages("circular")
```

** load remotes package and download rB3 from github **

SET SYSTEM TIMEZONE TO UTC! why?..

..the POSIXct date format used by rB3 can play havoc with your data editing if timezones are not handled well..

..so it can help to set your system environment to UTC, which avoids issues with tz offsets

File In

csv2rB3() - Import a raw dataset (csv)

Import a starting .csv file, which will be converted into a list of data frames:

1. the raw data block from your csv file; 'srcDF'
2. a copy of the raw data block, to be quality controlled; 'qcDF'
3. a matrix with similar dimensions to 1 & 2, to store qc action logID values; 'logDF'
4. a list of logID values and their meanings; 'logKey'
5. sensor/time-series metadata and control values used for filtering/plotting etc 'ctrls'
6. site/station meta data 'metaD'

For the documentation, run following code ?csv2rB3

Intial csv header rows can contain time-series/sensor metadata to be used in later functions (loaded as 'ctrls' DF within list). Row prior to start of data will be data frame headers

Date format must be yyyy-mm-dd hh:mm:ss, with header "DateTime"

Setting wd

Hide

```
setwd("C:/Users/km-admin/Dropbox/Git/rB3_wd/wd")
```

Hide

```
rB3demo <- csv2rB3("rB3demo_201507-201806_RAW_R.csv","Lake_Rotoehu",-38.5, 176.5,"NZ")
```

call the components of the rB3 object on the fly (not needed for rB3 operations).

Hide

```
names(rB3demo)
```

```
[1] "srcDF"  "qcDF"   "logDF"  "logKey" "ctrls"  "metaD"
```

srcDF: the unmodified data (data frame)

qcDF: Current version of the data in data frame (same size as the srcDF)

logDF: a 'log' file with similar structure to srcDF and qcDF, to store QC operations log (same size as the srcDF)

logKey: control key with logIDs and explanatory columns

ctrls: controls, extracted from extra header rows in the raw csv files ('ctrls')

metaD: site metadata, create from additional input args ('metaD')

You can access a data frame object by following syntax e.g.,

Hide

```
testDF <- rB3demo[["qcDF"]]
```

Explore and reformat your data

ShinyrB3() - GUI (graphic user interphase)

This module lets you investigate your data interactively using shiny package

For the documentation, run following code ?shinyrB3

Hide

```
shinyrB3(rB3demo)
```

Note that shiny occupies R studio so you need to shut the Shiny window in order to action any more commands..

More future updates will come around this GUI

Selecting variables in rB3

Variables can be called using key phrases/characters (i.e., all vars containing key word will be selected)

create vector of key phrases, for later functions, e.g.;

Hide

```
wqVars <- c('Fl','Tur','pH','DO')
```

?rB3getVars

Hide

```
rB3getVars(rB3demo, wqVars)
```

```
[1] "TmpDOs.d00050" "TmpDOs.d01000" "DOpsat.d00050" "DOpsat.d01000" "DOconc.d00050" "DOconc.d01000"
[7] "pHisft.d00050" "FlChlr.d00050" "FlPhyc.d00050"
```

retrieve varNames; 'All' (default) or select by keyphrase

?rB3getVars

Hide

```
rB3getVars(rB3demo, 'All')
```

```
 [1] "TmpWtr.d00050" "TmpWtr.d00150" "TmpWtr.d00300" "TmpWtr.d00500" "TmpWtr.d00700" "TmpWtr.d00900"
 [7] "TmpWtr.d01050" "TmpDOs.d00050" "TmpDOs.d01000" "DOpsat.d00050" "DOpsat.d01000" "DOconc.d00050"
[13] "DOconc.d01000" "pHisft.d00050" "FlChlr.d00050" "FlPhyc.d00050" "RadSWD.h00150" "TmpAir.h00150"
[19] "HumRel.h00150" "PrBaro.h00150" "WndSpd.h00150" "WndDir.h00150" "PpRain.h00150"
```

rB3stdze() - data trimming and temporal aggregation

Trim and standardize time intervals of a data frame

Our demo rawDF has 3 yrs data, some with 5 min data, some 15 min.

So let's trim dataset to **most recent 2 years**, and **aggregate to common (15 min) timestep**, using aggregation methods specific to each column as defined in the header metadata (ctrls$methodAgg).

Hide

```
rB3demo[["ctrls"]]$methodAgg
```

```
 [1] "mean"   "mean"   "mean"   "mean"   "mean"   "mean"   "mean"   "mean"   "mean"

[10] "mean"   "mean"   "mean"   "mean"   "mean"   "mean"   "mean"   "median" "mean"

[19] "mean"   "mean"   "mean"   "circular" "sum"
```

For example, here we'll aggregate by mean, but sum for rainfall, and circular averaging for wind direction

?rB3stdze

Hide

```
# aggregate the data to 15 min timestep - can take a while on big DFs!
rB3agg <- rB3stdze(rB3in = rB3demo,
                   varNames = 'All',
                   startDate = '2016-07-01',
                   endDate = '2018-06-30 23:45:00',
                   timestep = 15,
                   aggAll = FALSE)
```

varWrangle() - create, remove or move variables

?varWrangle

Hide

```
# add variable(s) after 3rd variable (3rd excluding DateTime)
rB3agg <- varWrangle(rB3agg,
                     varNames = "TESTVAR",
                     task = "add",
                     loc = 4)
```

```
[1] "TESTVAR added to the rB3 obj"
```

Hide

```
rB3getVars(rB3agg)
```

```
 [1] "TmpWtr.d00050" "TmpWtr.d00150" "TmpWtr.d00300" "TmpWtr.d00500" "TESTVAR"       "TmpWtr.d00700"
 [7] "TmpWtr.d00900" "TmpWtr.d01050" "TmpDOs.d00050" "TmpDOs.d01000" "DOpsat.d00050" "DOpsat.d01000"
[13] "DOconc.d00050" "DOconc.d01000" "pHisft.d00050" "FlChlr.d00050" "FlPhyc.d00050" "RadSWD.h00150"
[19] "TmpAir.h00150" "HumRel.h00150" "PrBaro.h00150" "WndSpd.h00150" "WndDir.h00150" "PpRain.h00150"
```

Hide

```
# remove variable(s) by "keyword" TESTVAR
rB3agg <- varWrangle(rB3agg,
                     varNames = "TESTVAR",
                     task = "rm")
```

```
[1] "TESTVAR removed from the rB3 obj"
```

Hide

```
rB3getVars(rB3agg)
```

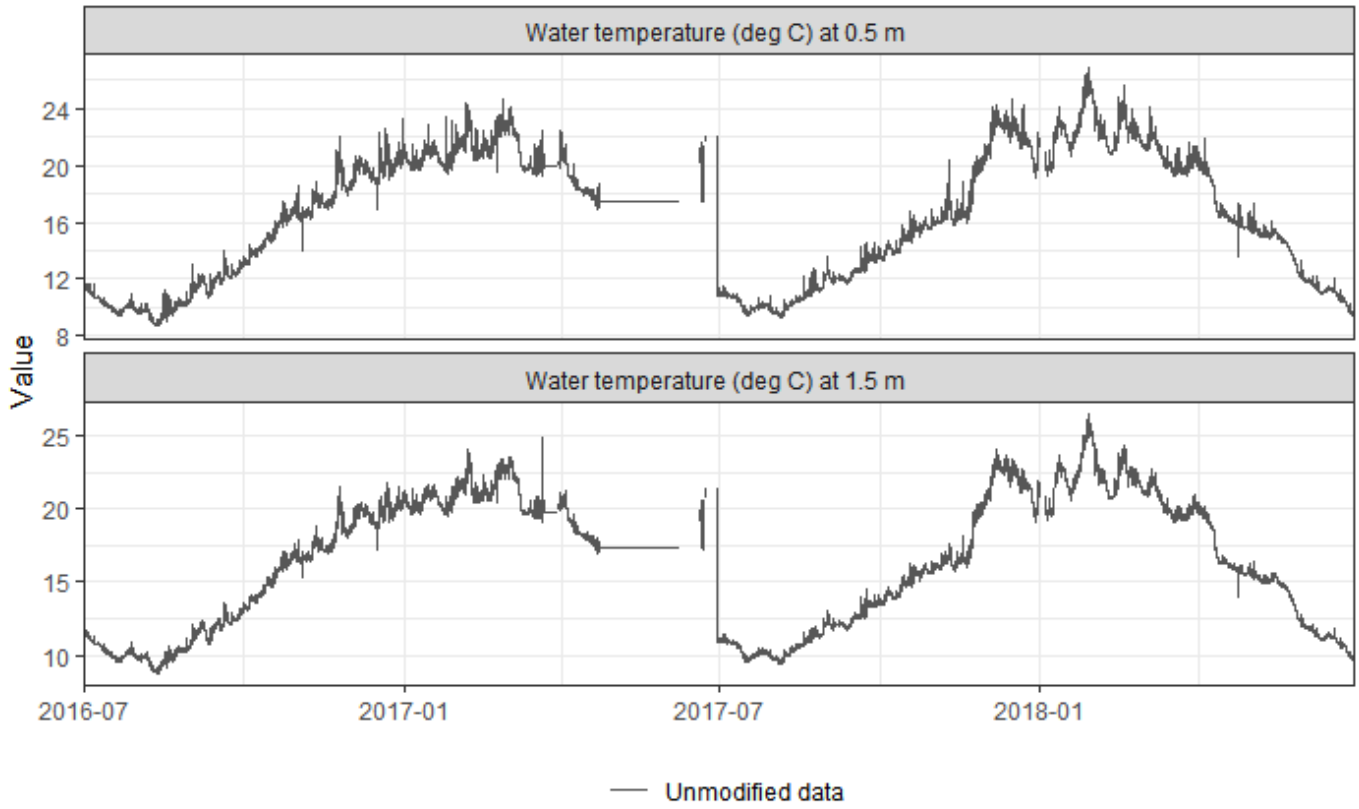
```
 [1] "TmpWtr.d00050" "TmpWtr.d00150" "TmpWtr.d00300" "TmpWtr.d00500" "TmpWtr.d00700" "TmpWtr.d00900"
 [7] "TmpWtr.d01050" "TmpDOs.d00050" "TmpDOs.d01000" "DOpsat.d00050" "DOpsat.d01000" "DOconc.d00050"
[13] "DOconc.d01000" "pHisft.d00050" "FlChlr.d00050" "FlPhyc.d00050" "RadSWD.h00150" "TmpAir.h00150"
[19] "HumRel.h00150" "PrBaro.h00150" "WndSpd.h00150" "WndDir.h00150" "PpRain.h00150"
```

gg_facetVar() - Basic panel plots

?rB3gg

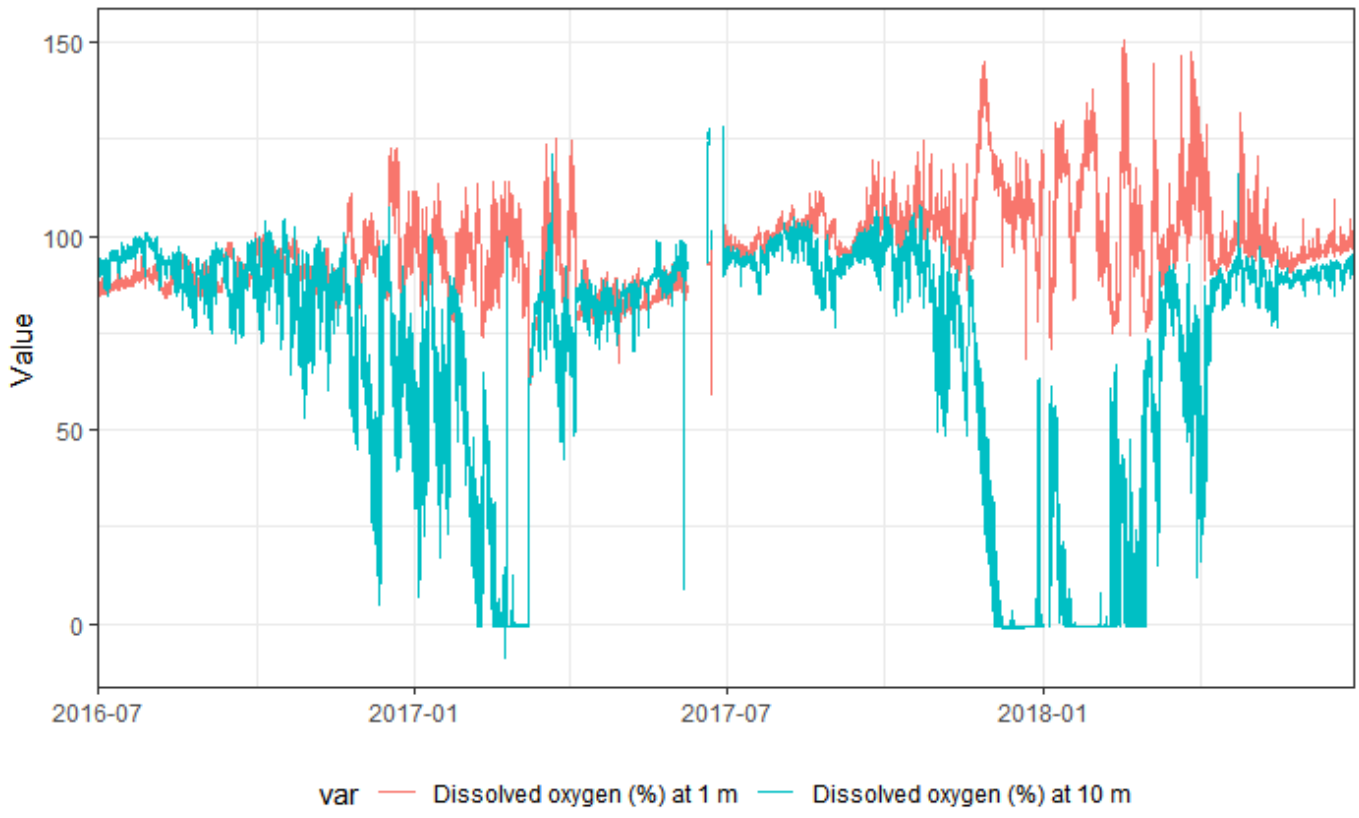
Hide

```
# plot the variables called by the the keywords, saved to figures dir
rB3gg(rB3in = rB3agg,
      varNames = c("TmpWtr.d00050","TmpWtr.d00150"),
      srcColour = 'grey34',
      facet = TRUE,
      showPlot = TRUE)
```



Hide

```
rB3gg(rB3in = rB3agg,
      varNames = 'DOpsat',
      srcColour = 'grey34',
      facet = FALSE,
      showPlot = TRUE ,
      savePlot = 'figures/RAW_WQ_',
      dpi = 400)
```



Backup the aggregated data frame, in case we want to revert later

Hide

```
rB3agg2 <- rB3agg
```

Hide

```
shinyrB3(rB3agg2)
```

assignVal() - Delete or change selected data values

This function replace values in specified regions of data with a numerical value or with NA

?assignVal

Select a region from your shiny plot containing erroneous data, then paste the example function, e.g.:

Hide

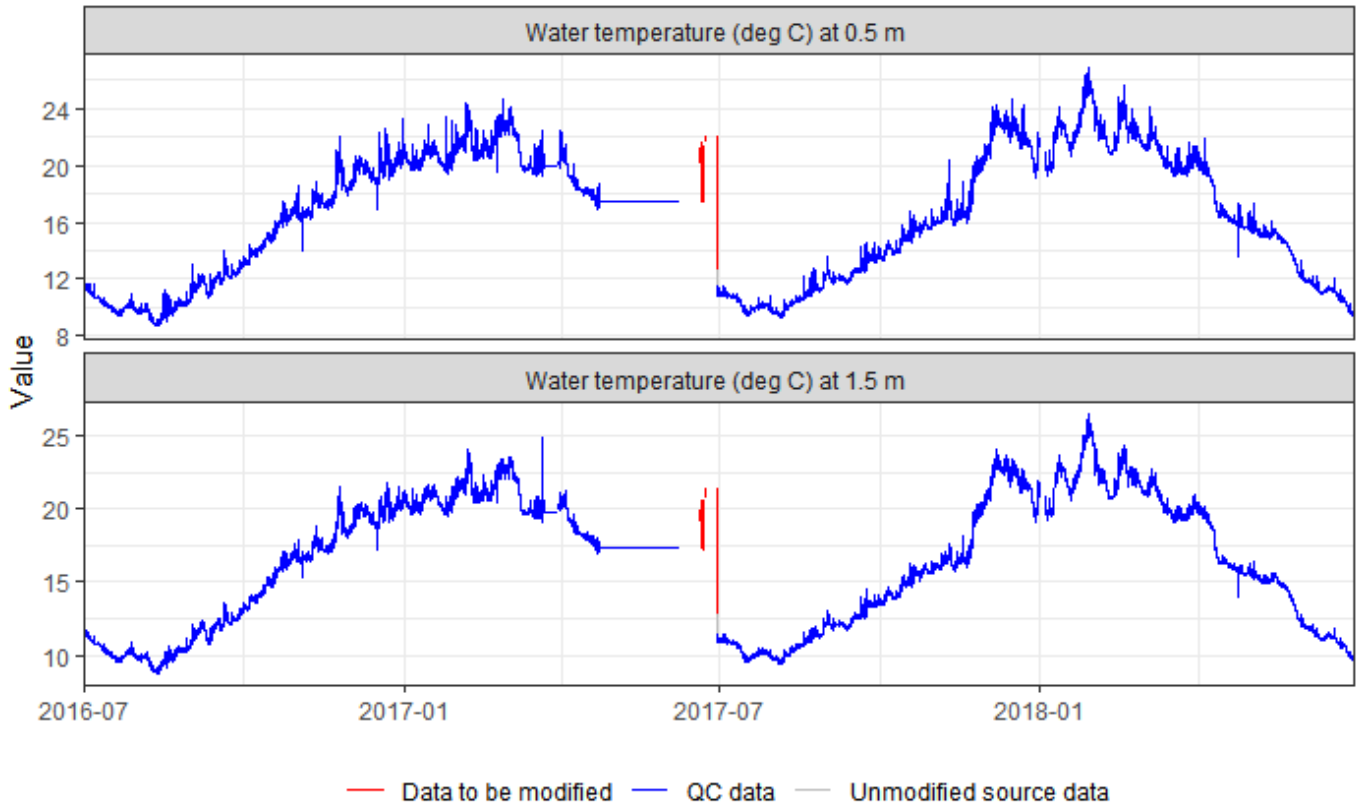
```
rB3agg2 <- assignVal(rB3agg2,
                     varNames = c('TmpWtr.d00050','TmpWtr.d00150'),
                     startDate = "2017-06-15 23:05:14",
                     endDate = "2017-07-06 11:00:38",
                     minVal = 12,
                     maxVal = 22.9,
                     newVal = NA,
                     logID = "Shiny",
                     Reason = "Manual removal",
                     showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
1
```

```
[1] "Changes have been applied"
```



Apply filters using 'ctrls' values

filterRoc() - Filter data by rate of change

Replace values exceedign specified rate of change with NA

?filterRoc

Hide

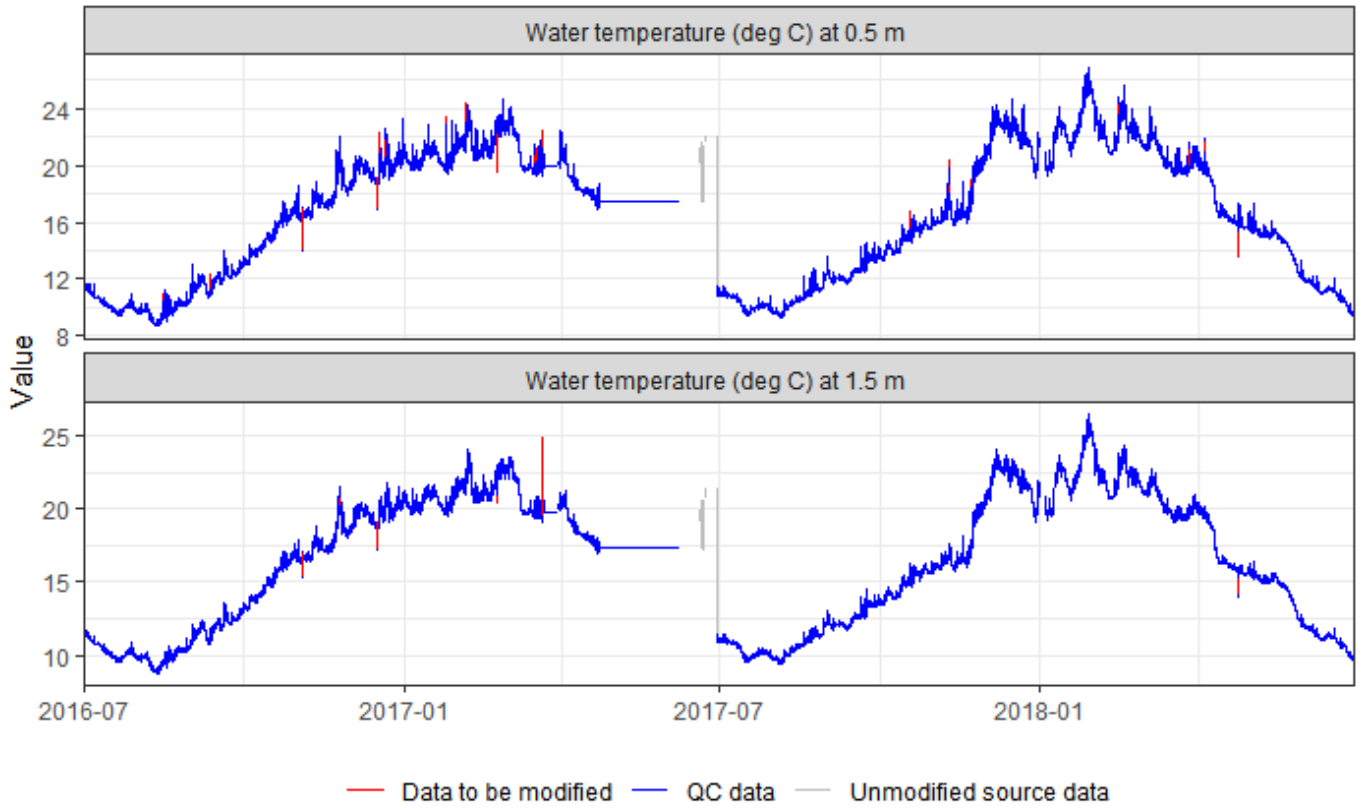
```
rB3agg2 <- filterRoc(rB3agg2,
                     varNames = c('TmpWtr.d00050','TmpWtr.d00150'),
                     maxRoc = 0.5,
                     showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
1
```

```
[1] "Changes have been applied"
```



If showPlot is TRUE, so you must enter your choice (1 = accept, 2 = decline) to continue

filterReps() - filter by repeated values

Replace data where identical value has been repeated more than n = maxReps

?filterReps

Hide

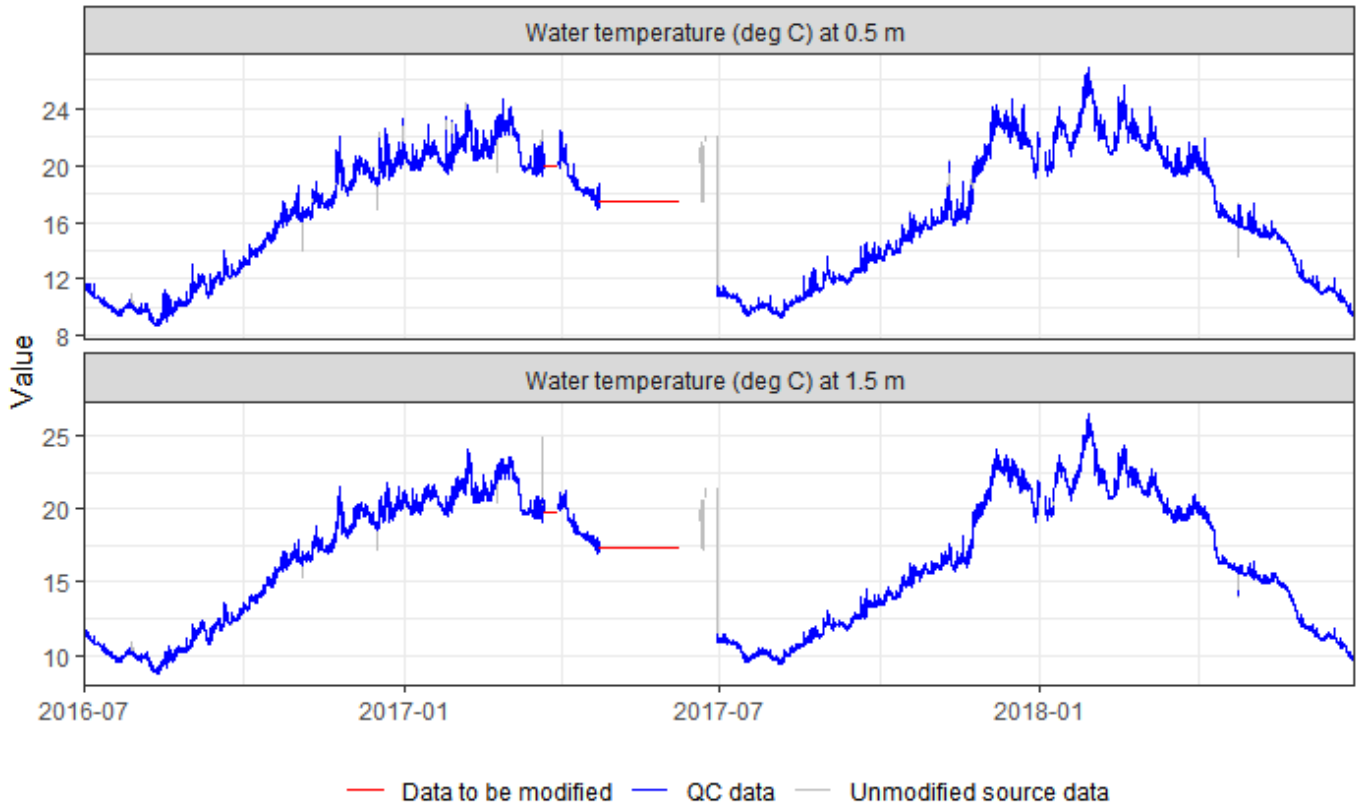
```
rB3agg2 <- filterReps(rB3agg2,
                      varNames = c('TmpWtr.d00050','TmpWtr.d00150'),
                      maxReps = 20,
                      showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
1
```

```
[1] "Changes have been applied"
```



filterMinMax() - limit data value by range

Filter data below minVal or above maxVal (either specified, or from 'ctrls'/headers)

?filterMinMax

Hide

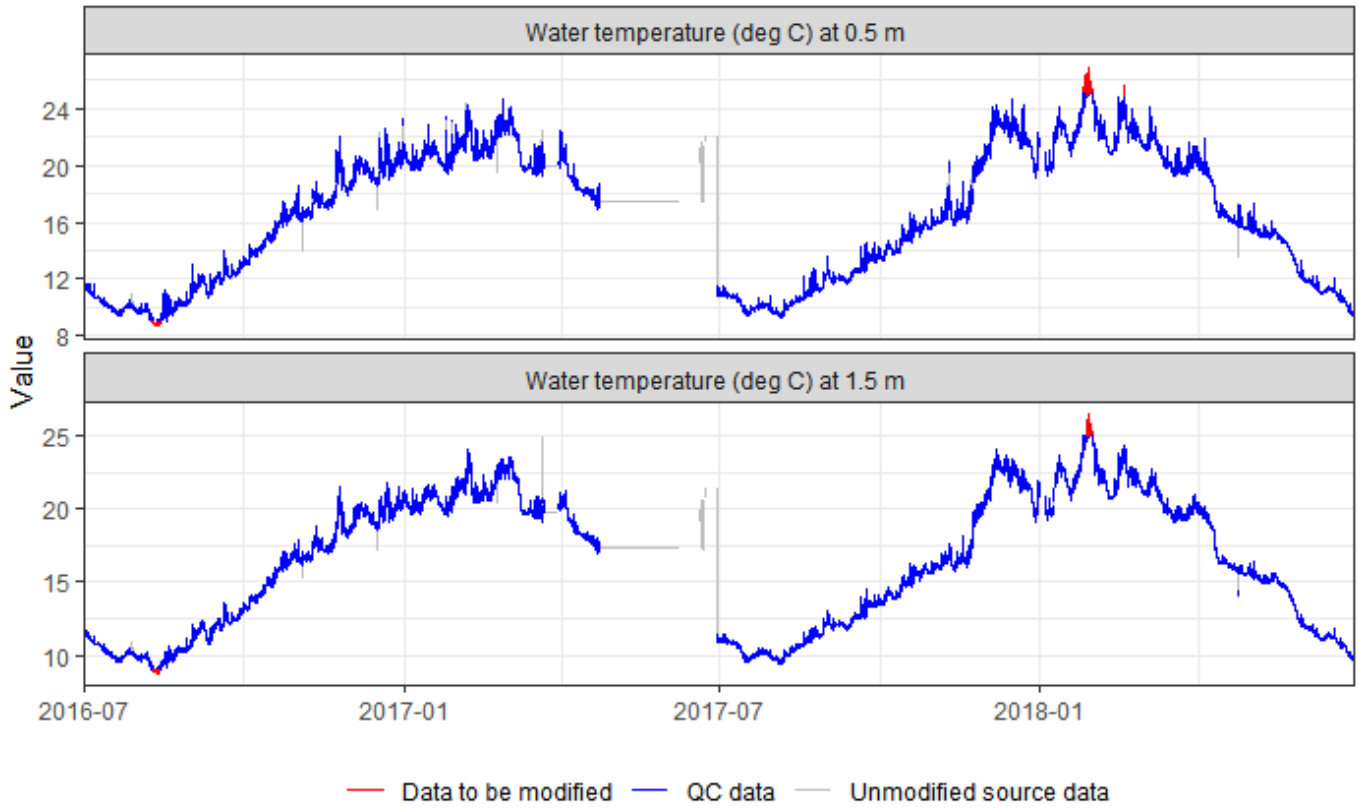
```
rB3agg2 <- filterMinMax(rB3agg2,
                         varNames = c('TmpWtr.d00050','TmpWtr.d00150'),
                        filterMin = 9,
                        filterMax = 25,
                        showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
1
```

```
[1] "Changes have been applied"
```



applyInterp() - linearly interpolate NA values

?applyInterp

Hide

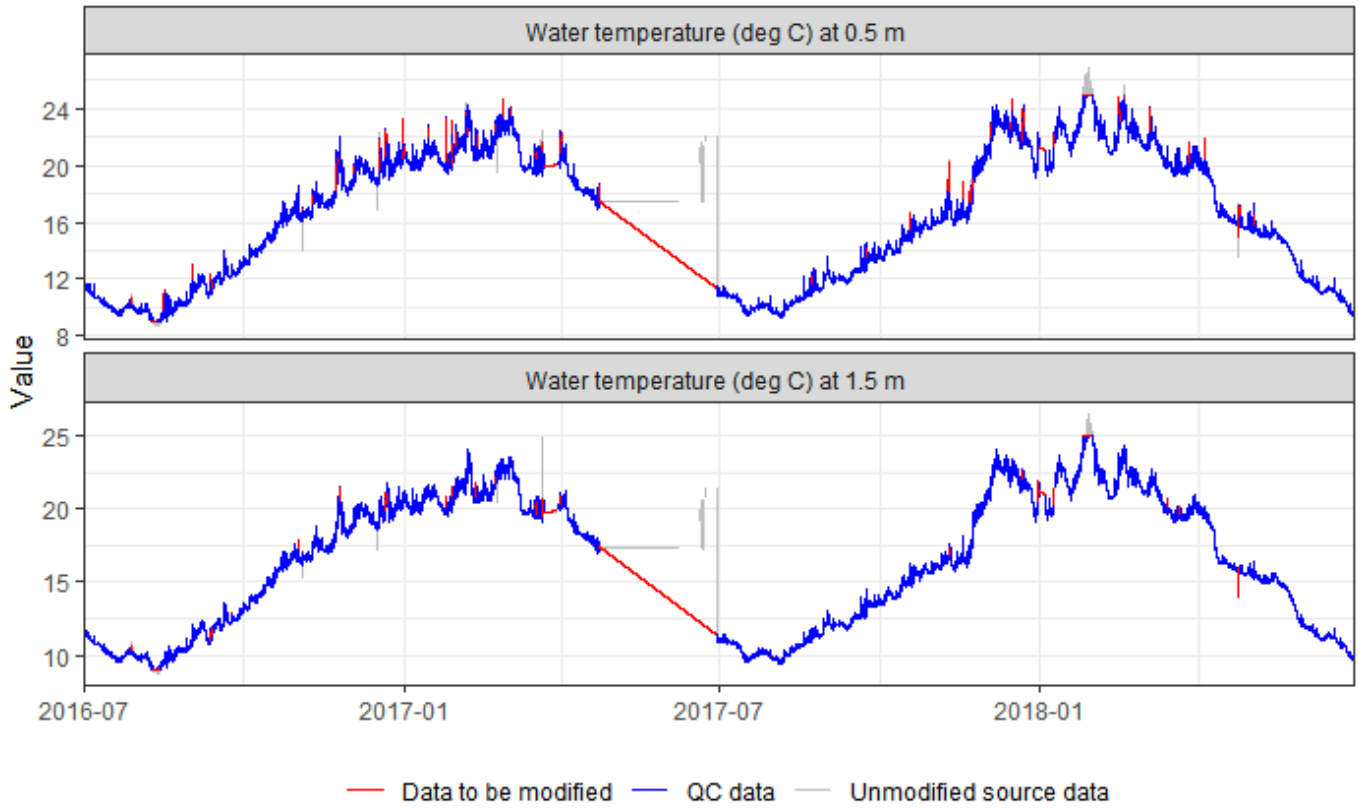
```
rB3agg2 <- applyInterp(rB3agg2,
                       varNames = c('TmpWtr.d00050','TmpWtr.d00150'),
                       showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
1
```

```
[1] "Changes have been applied"
```



Visualise your rB3 process

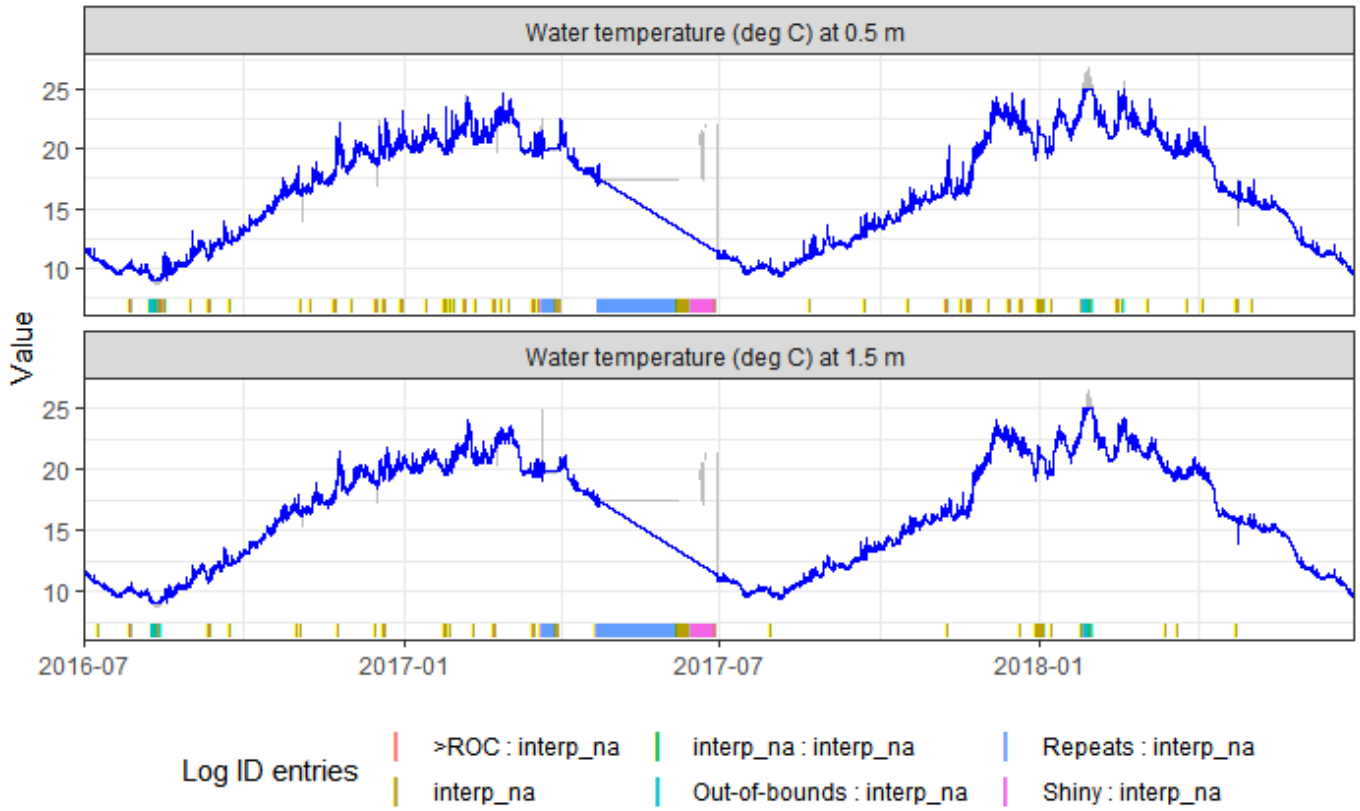
logsPlot() - visualise source, QC data and modifications log

?logsPlot

visualise changes to data

Hide

```
logsPlot(rB3in = rB3agg2,
         varNames = c('TmpWtr.d00050','TmpWtr.d00150'),
         srcColour = 'grey')
```



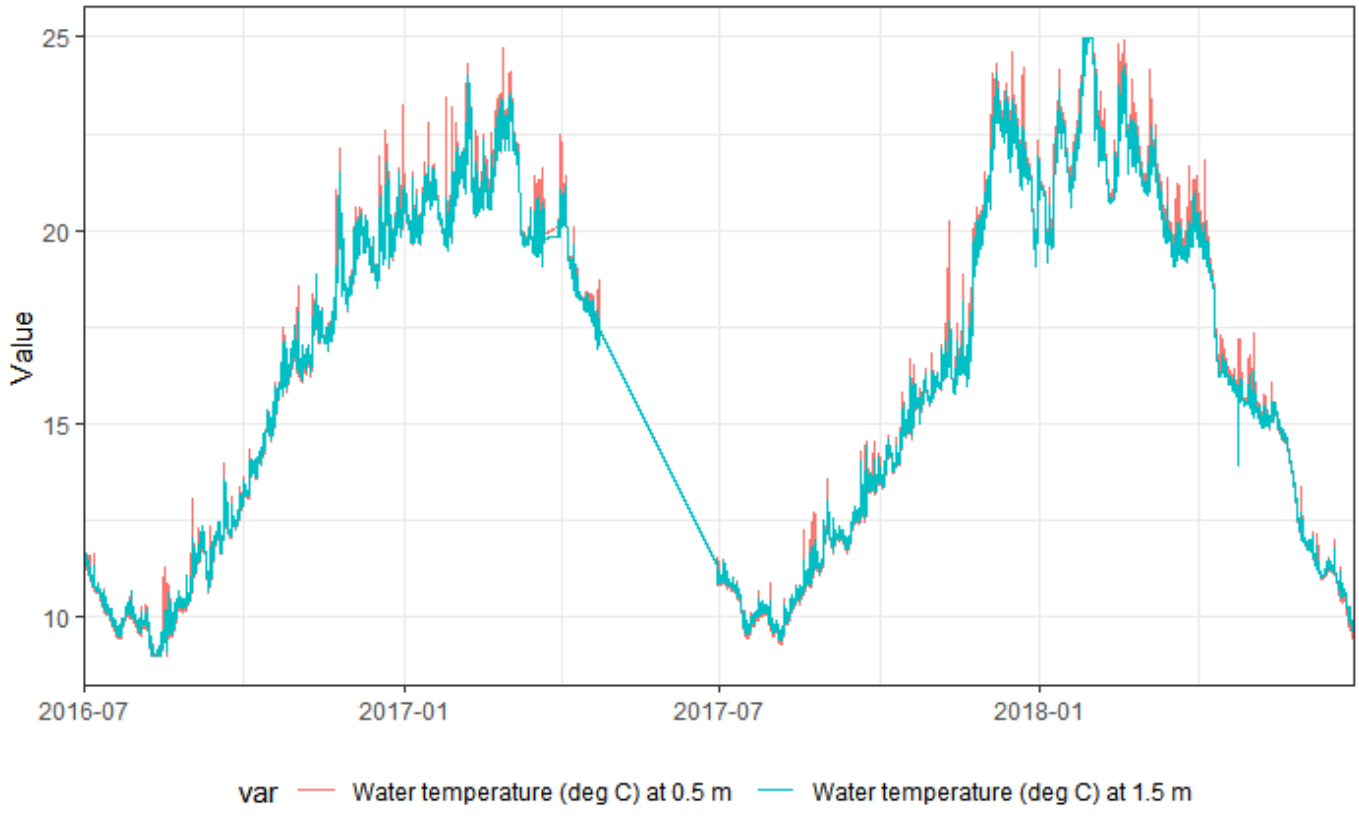
?rB3gg

View the final before and after, without logs

Hide

```
rB3gg(rB3in = rB3agg2,
      varNames = c('TmpWtr.d00050','TmpWtr.d00150'),
      srcColour = 'orange',
      qcColour = 'blue') #, savePlot = 'figures/RAW_WQ_',  dpi = 400)
```

```
[1] "Only quality controlled data are displayed for non-faceted plots"
```



rB3export() EXPORTING rB3 DATA

Export data from the rB3 object into csv files

?rB3export

Hide

```
rB3export(rB3agg2,
          varNames = 'All',
          qc = T,
          src = T,
          metadata = T)
```

Hide

```
rB3agg3 <- rB3agg2
```

Advanced QA/QC function

applyNth()

Apply a mathematical transformation, e.g. new = a + b(old) + c(old)^2 + c(old)^3 + …etc

?applyNth

Hide

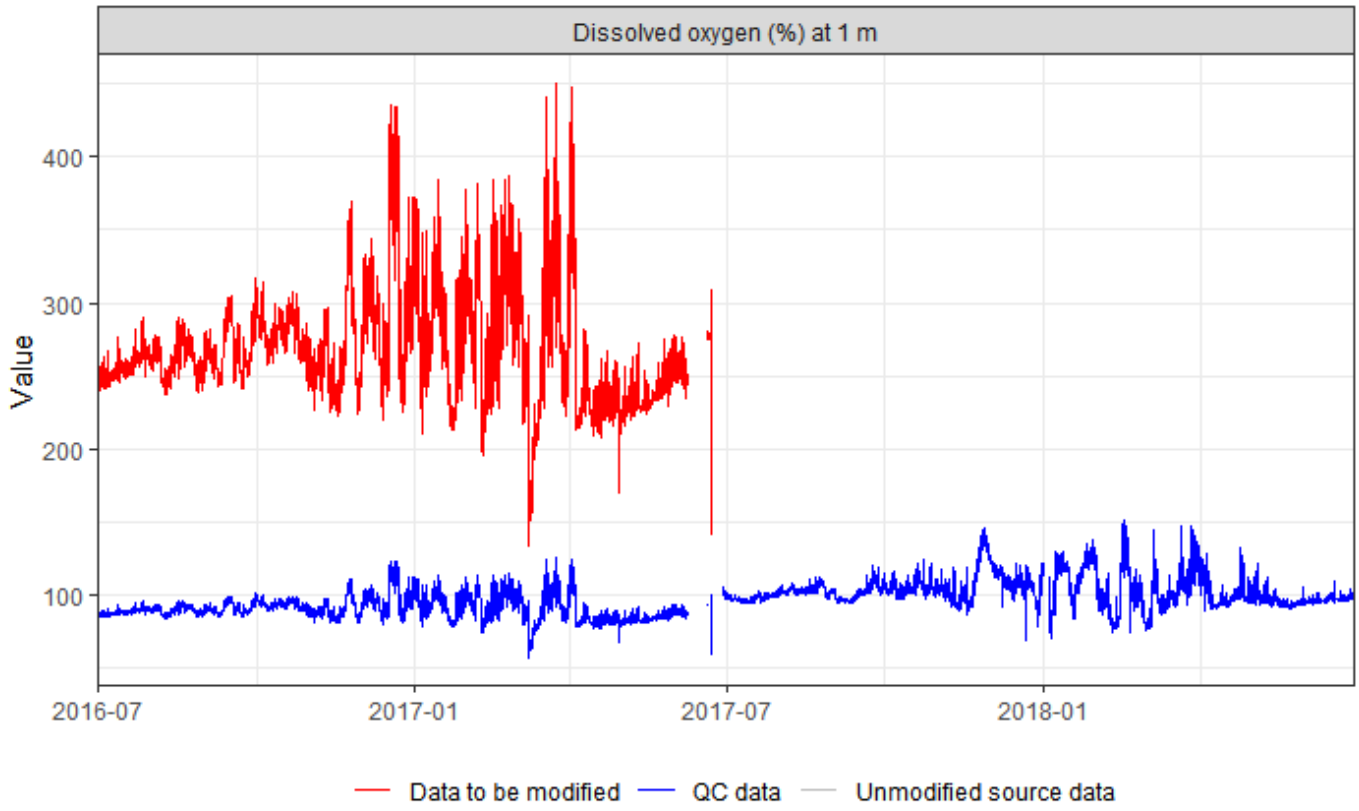
```
rB3agg2 <- applyNth(rB3in = rB3agg2,
                    startDate = '2016-07-01 00:00:00',
                    endDate = '2017-06-28 23:45:00',
                    varNames = 'DOpsat.d00050',
                    coeffs = c(12,1,0.02),
                    showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
2
```

```
[1] "Changes were not applied"
```



driftCorr()

Correct linear sensor drift (assumes consistent timestep)

?driftCorr

Hide

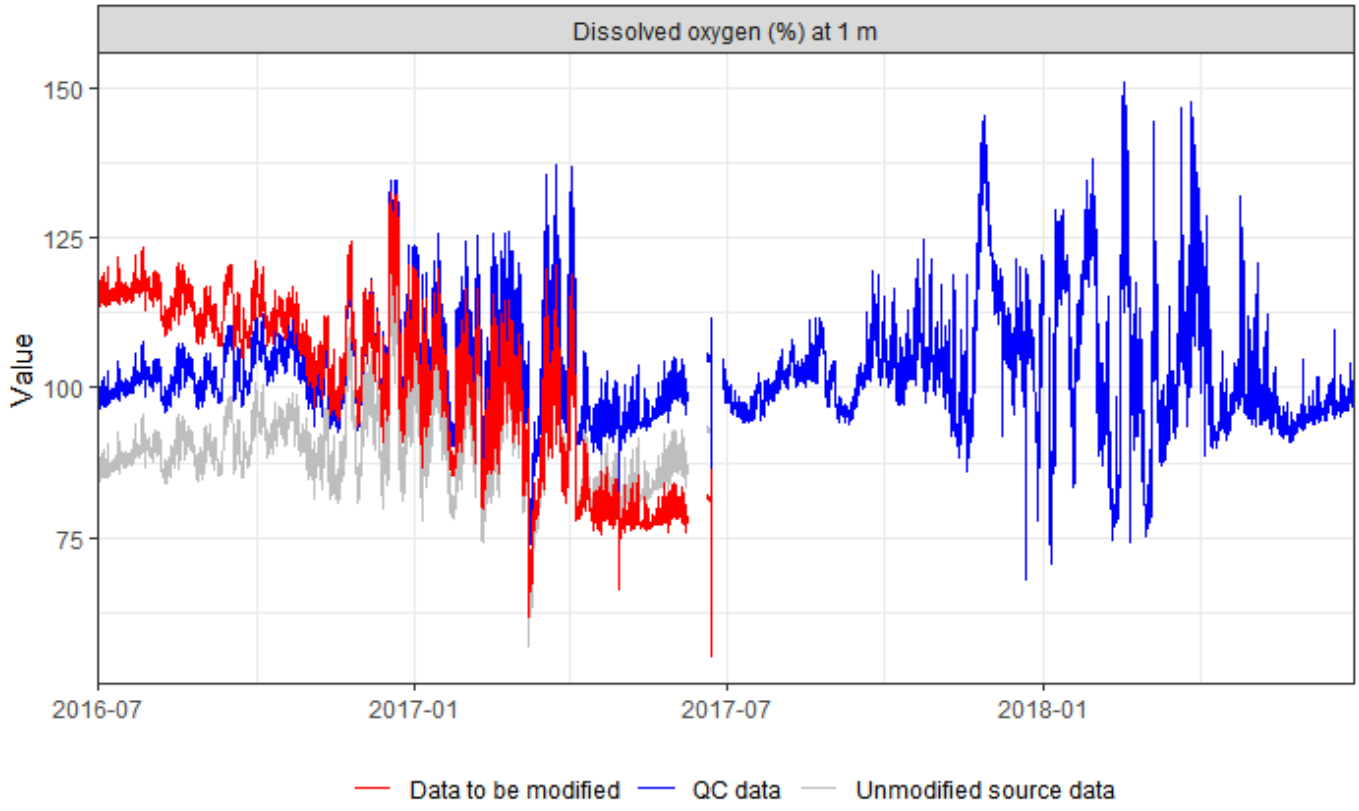
```
rB3agg2 <- driftCorr(rB3agg2,
                     '2016-07-01 00:00:00',
                     '2017-06-28 23:45:00',
                     'DOpsat.d00050',
                     lowRef = 0,
                     lowStart =  0,
                     lowEnd =  0,
                     highRef = 100,
                     highStart = 85,
                     highEnd = 130,
                     showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
1
```

```
[1] "Changes have been applied"
```



Some useful functions

tmprAlign()

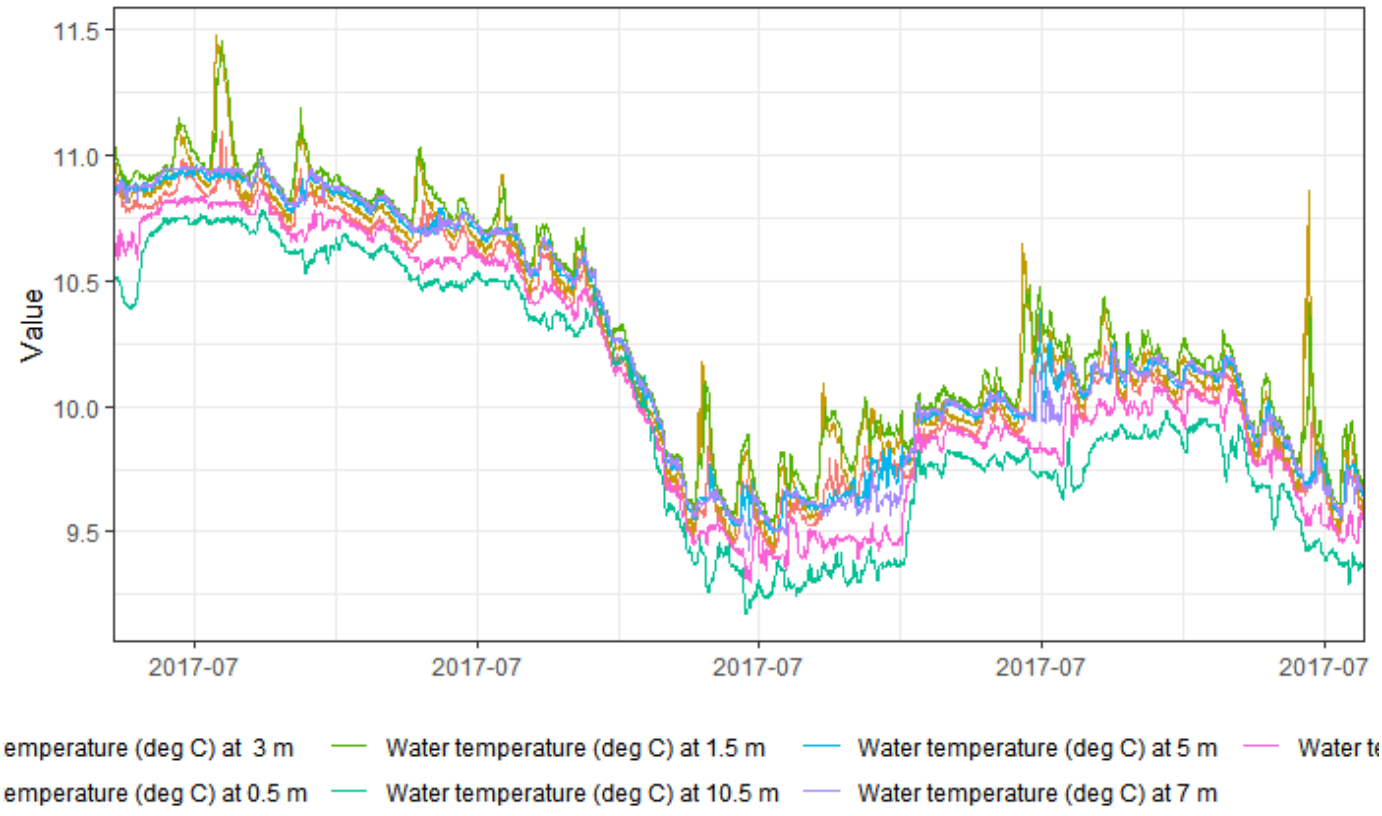
?tmprAlign

Post-calibrate temperature sensors based on periods of mixing, as found by temp differences and wind speed (optional)

Pre tmprAlign()

Hide

```
rB3gg(rB3in = rB3agg2,
      varNames = 'TmpWtr',
      startDate = '2017-07-01',
      endDate = '2017-08-01',
      facet = FALSE,
      showPlot = T)
```



Hide

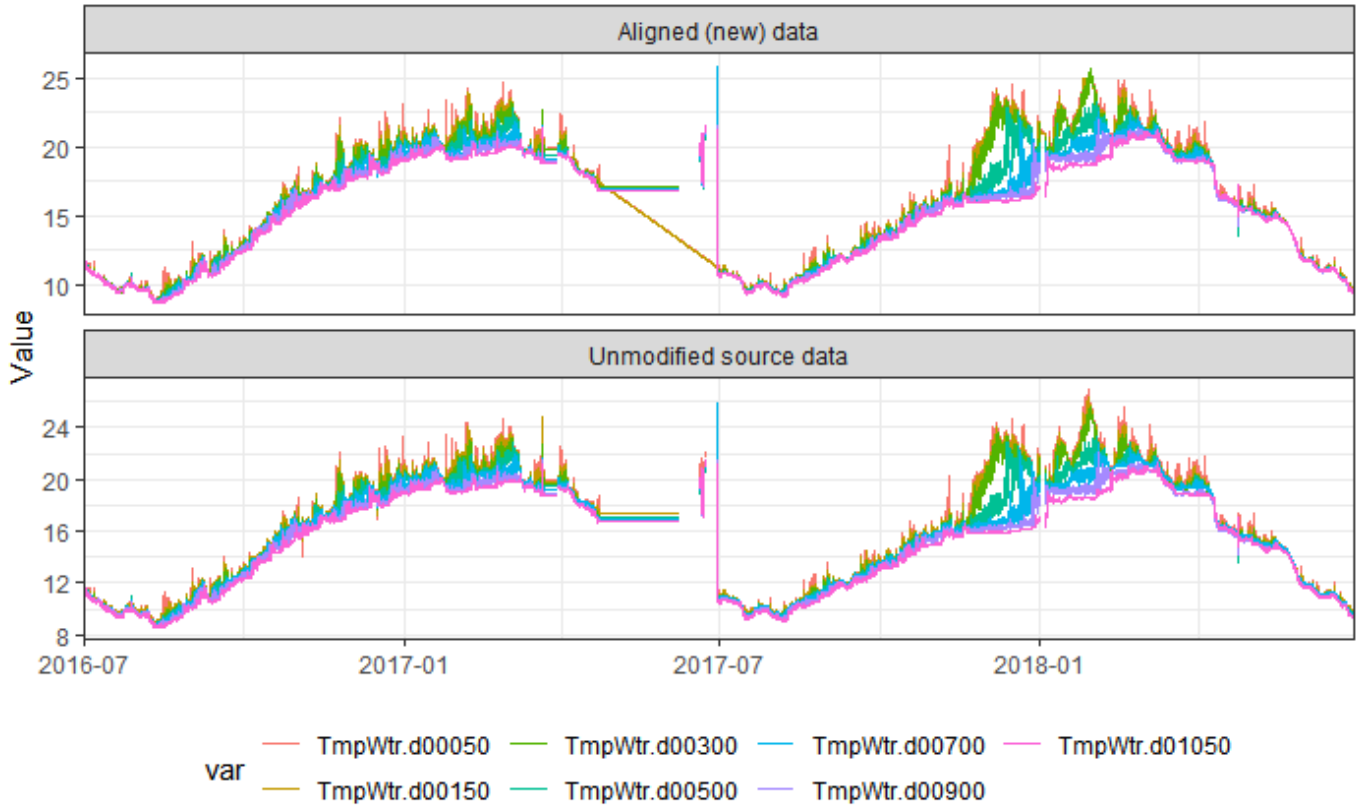
```
rB3agg2 <- tmprAlign(rB3agg2,
                     varNames = 'TmpWtr',
                     dTPerctile = 0.2,
                     logID = "tpmAlign",
                     Reason = "Interp",
                     showPlot = T,
                     plotType = 'All')
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
1
```

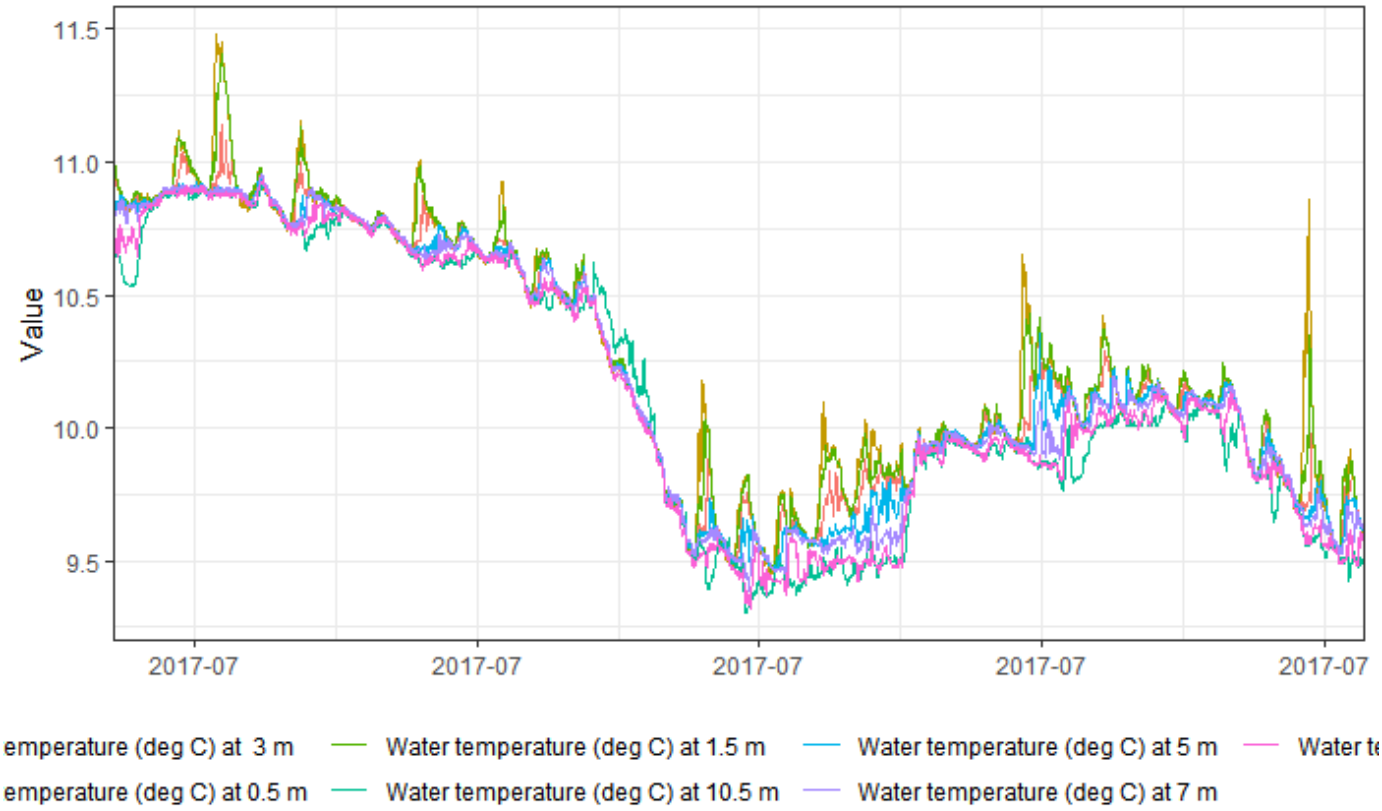
```
[1] "Changes have been applied"
```



Post tmprAlign()

Hide

```
rB3gg(rB3in = rB3agg2,
      varNames = 'TmpWtr',
      startDate = '2017-07-01',
      endDate = '2017-08-01',
      facet = FALSE,
      showPlot = T)
```



FUNCrB3() - apply your own equation

?FUNCrB3 Apply a custom function using rB3

Simple example:

multiply a variable by 2

Hide

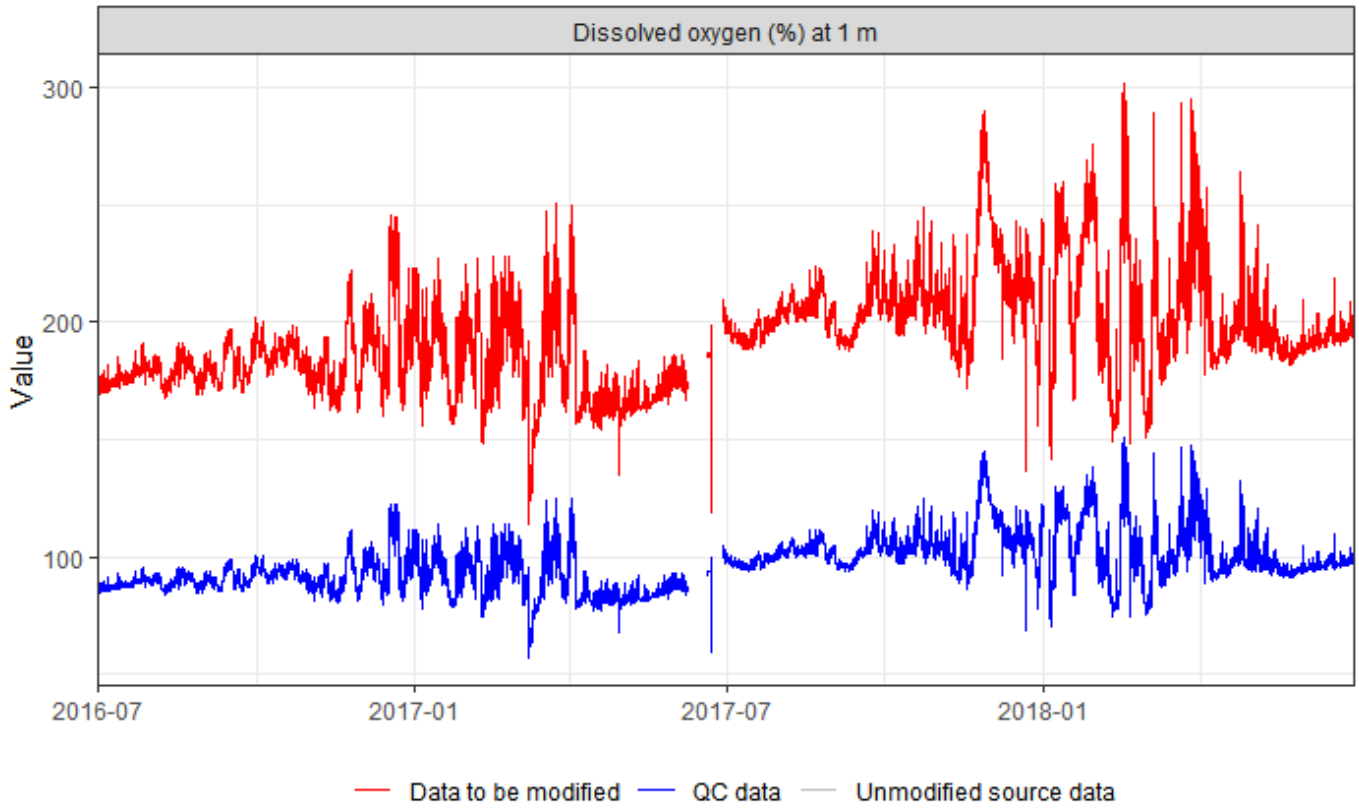
```
# define a simple function ( result = input variable * 2)
test <- function(eqnVars) {eqnVars[1] * 2}
# apply this custom function
rB3agg2 <- FUNCrB3(rB3agg2,
                   varNames = 'DOpsat.d00050',
                   eqnVars = 'DOpsat.d00050',
                   FUN = test,
                   showPlot = T)
```

```
Apply these changes?

1: Yes
2: No
```

Hide

```
2
```

```
[1] "Changes were not applied"
```



Complex example:

calculate DO (mg/L) using (DO (%sat) and water temperature) using USGS method..

Meyers, D.N. (2011) https://water.usgs.gov/admin/memo/QW/qw11.03.pdf (https://water.usgs.gov/admin/memo/QW/qw11.03.pdf*)

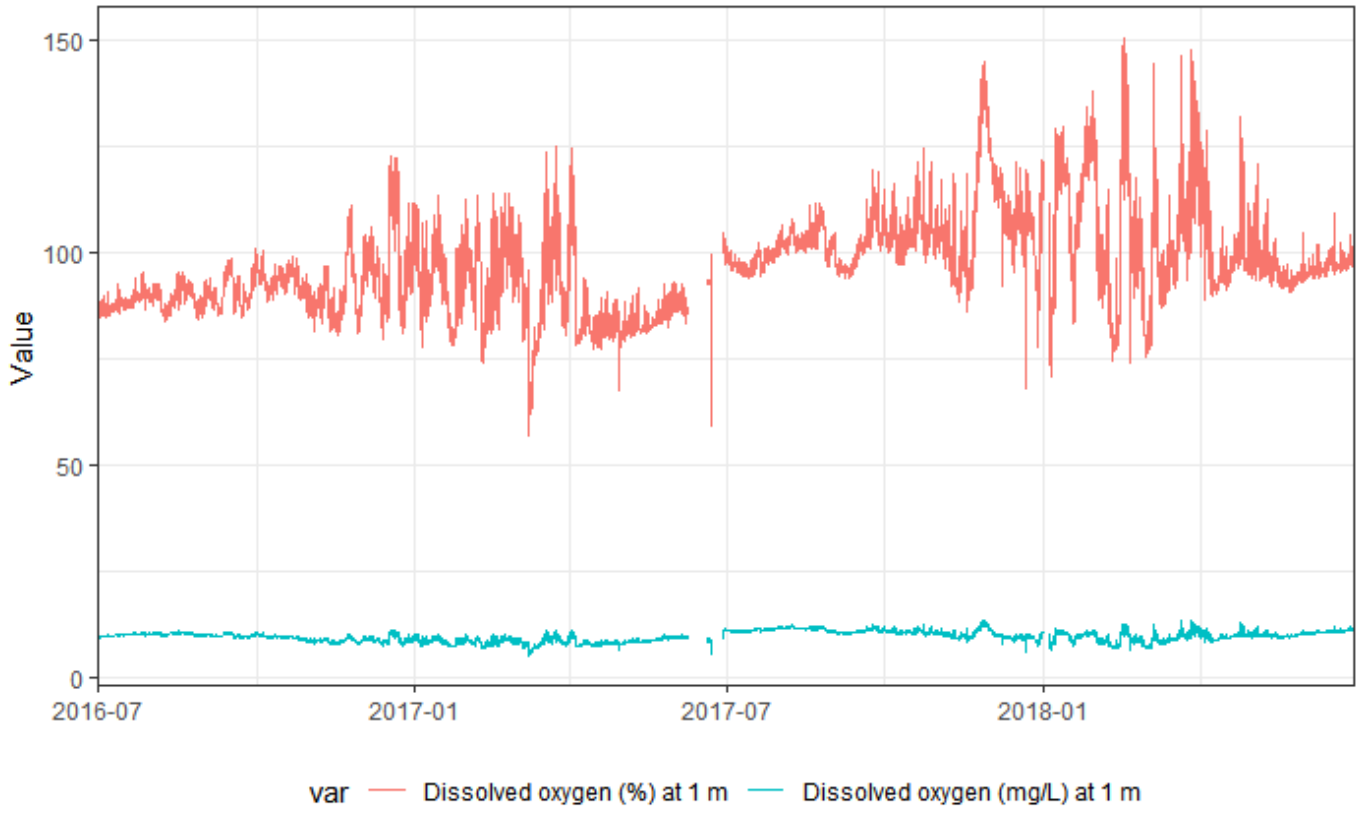
eqn: DOmg = (exp(-139.34411 + ((157570.1*(1/(tmpwtr +273.15))) + (-66423080*((1/(tmpwtr +273.15))^2)) + (12438000000*((1/(tmpwtr +273.15))^3)) + (-862194900000*((1/(tmpwtr +273.15))^4))))) * DOsat *0.01

..where tmpwtr = water temperature and DOsat = dissolved oxygen saturation

Hide

```
rB3gg(rB3agg2,
      varNames = c('DOpsat.d00050','DOconc.d00050'),
      showPlot = T,
      srcColour = 'orange',
      qcColour = 'blue')
```

```
[1] "Only quality controlled data are displayed for non-faceted plots"
```



Hide

```
# define the list of input variables required for the calculation
eqnVars = c('TmpWtr.d00050','DOpsat.d00050')
# define the function, using eqnVars[1] - tmpwtr and eqnVars[2] = DOsat
DOsat2mg <- function(eqnVars) {
  (exp(-139.34411 + ((157570.1*(1/( eqnVars[1] +273.15))) +
      (-66423080*((1/( eqnVars[1] +273.15))^2)) +
        (12438000000*((1/( eqnVars[1] +273.15))^3)) +
           (-862194900000*((1/( eqnVars[1] +273.15))^4))))) * eqnVars[2] * 0.01
}
```

Hide

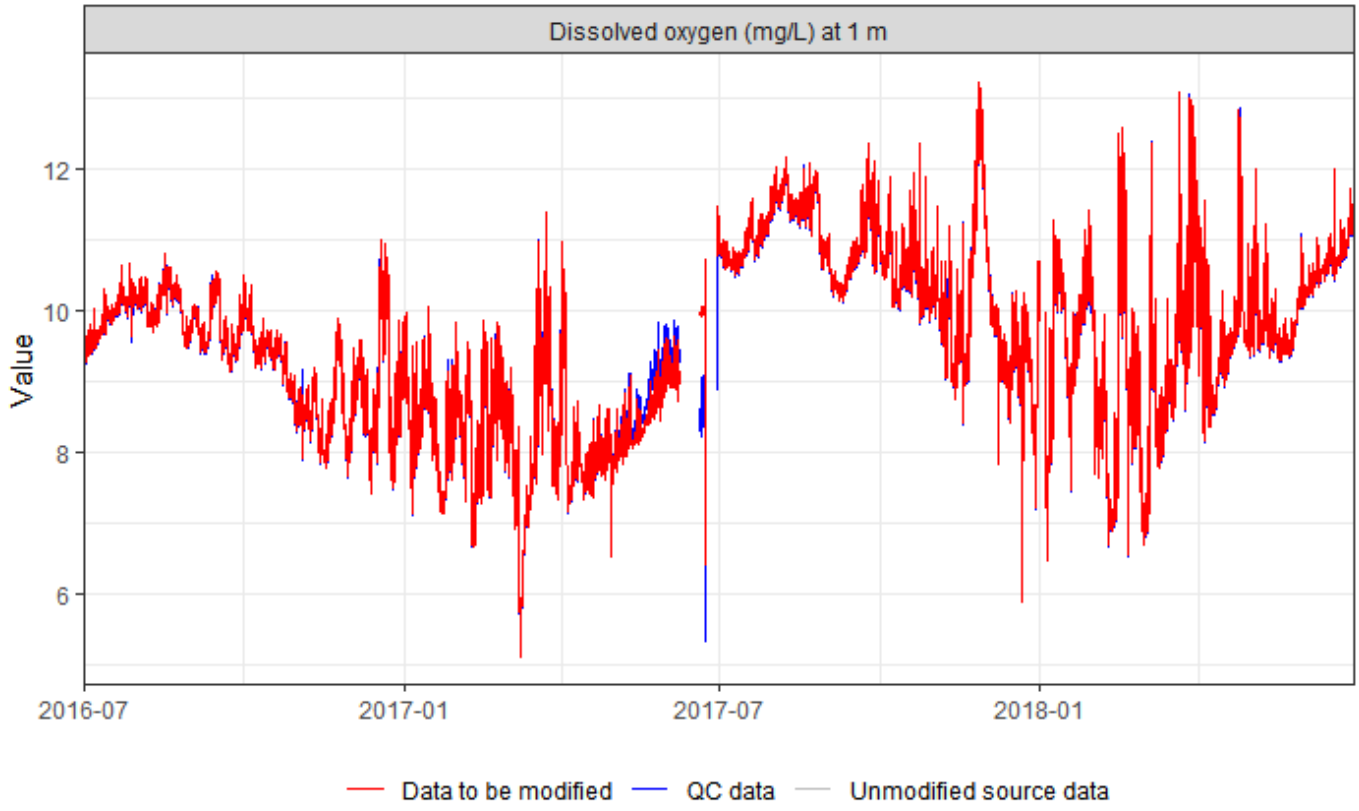
```
rB3agg2 <- FUNCrB3(rB3agg2,
                   varNames = 'DOconc.d00050',
                   eqnVars = eqnVars,
                   FUN = DOsat2mg,
                   showPlot = T)
```

```
Apply these changes?

1: Yes
2: No

Enter an item from the menu, or 0 to exit
```

Hide

```
1
```

```
[1] "Changes have been applied"
```



Export files as Lake Analyzer inputs

writeLAinputs() -

write cleaned up temperature and wind data to .wtr and .wnd files for direct input to rLakeAnalyzer

?writeLAinputs

Hide

```
writeLAinputs(rB3in = rB3agg2,
              wtrNames = 'TmpWtr',
              wndName = 'WndSpd',
              wndHeight = 1.5)
```