In-house R Packages for BoPRC

James Dare

7th March 2018

Introduction

Hopefully you all have a reasonable understanding of the functionality of R following the course run by Martin Bader. One of the most useful things from our perspective at BoPRC, is the ability to link to different databases and analyse data directly from them, without the need to load pre-extracted CSV or Excel files. This is preferable, from a data management point of view, as it means that scientists are always referring to the 'source of truth', which is constantly being updated, quality checked, and improved. Eventually this will negate the need to have 'pet databases' stored on hard drives, as data will become accessible at the click of a button (or a line or two of code).

In addition to this, Darren and I are working hard to build packages of functions that may be useful for everyday council analyses. With proper training, these packages should save you a lot of time as R can do most of the mundane data crunching for you!

The aim of this tutorial is to introduce you to two packages: 'Aquarius' built by Data Services EDS. JobRequests@boprc.govt.nz, and 'BoPRC' built by james.dare@boprc.govt.nz. Please feel free to contact us if you have any questions, or more importantly, any suggestions for future improvements (including new analysis tools).

Downloading and Updating Packages

The two in-house packages can be downloaded using the following lines of code.

For aquarius

```
la
st_packages = list.files("V:/Applications/Data Services/Data Services/
                             Environmental Data Services/Tools/Software/R")
la
st_package = la
st_packages[grep("aquarius_",
    la
st_packages)]
install.packages(pas
("V:/Applications/Data Services/Data Services/
                         Environmental Data Services/Tools/Software/R/",
    la
st_package, sep = ""), repos = NULL)
.rs.restartR()
```

For BoPRC

```
latest_packages = list.files("V:/Applications/Data Services/Data Services/
                             Environmental Data Services/Tools/Software/R")
latest_package = latest_packages[grep("BoPRC_", latest_packages)]
install.packages(paste("V:/Applications/Data Services/Data Services/
                         Environmental Data Services/Tools/Software/R/",
    latest_package, sep = ""), repos = NULL)
.rs.restartR()
```

As the packages are a work in progress and will be added to and improved over time, there will be a need to update periodically. We have made this easy with a function within the package. The function will look for the most up to date package and then overwrite the existing, allowing for any changes to be available immediately.

For aquarius

```
update_aquarius_package()
```

For BoPRC

```
update_boprc_package()
```

Additional Packages Required

You will also need to have access to the following external packages (and their dependents) to complete this tutorial: 'httr', 'stringr', 'ggplot2','grid','gridExtra', 'wq'.

The Aquarius Package

The Aquarius package includes functions that enable the user to link and extract data directly from our Aquarius database.

Before we can begin we need to load the aquarius package.

```
library(aquarius)
```

Searching for Site Information

We will explore the functionality of the Aquarius package by importing a basic flow record from a site of interest, let's say 'Whakatane at Whakatane (Valley Road)'. This function requires the siteID in order to know exactly what site we are talking about. Sometimes you may have this and can skip the next step, but in this case we will use the Aquarius package to search for the correct site ID which is housed in Aquarius (the database) tables.

Use the function 'searchlocationname()' to conduct a wildcard search for all sites that have 'Whakatane' in the title. There are quite a few, so it would pay to save this as a dataframe for easier reference.

```
WHKsites <- searchlocationname("*Whakatane*")
```

Use the 'view' function to view the object you just created, or click on the name 'WHKsites' in the Data pane on the right.

```
View(WHKsites)
```

You can click on the headers of the table to sort by any of the column headings, if that makes things easier.

You can see that the 31st row in the dataframe 'WHKSites' contains the site we are interested in. If you scroll across you can see that the 6th column contains the siteID that we require.

Site information is stored in a horizontal format, but we can use the R transpose function to make this more readable.

```r
# transpose data from the 31st row
t(WHKsites[31, ])
```

```
##                             31
## LOCATIONID                  "16851"
## LOCATIONFOLDERID            "89"
## LASTMODIFIED                "2017-12-08T09:39:11.993"
## LOCATIONNAME                "Whakatane at Whakatane (Valley Road)"
## DESCRIPTION                 NA
## IDENTIFIER                  "LL080607"
## LOCATIONTYPEID              "1000"
## LATITUDE                    "-38.00514"
## LONGITUDE                   "176.9951"
## SRID                        "4326"
## ELEVATIONUNITS              "m"
## ELEVATION                   "0"
## UTCOFFSET                   "12"
## TIMEZONE                    NA
## DEFAULTROLEID               NA
## AQUSERDATA_                 NA
## REGIONALMONITORINGSITE      "Current"
## AUTOMATED                   "Yes"
## SURFACEWATERLEVEL           "Yes"
## SURFACEWATERFLOW            "Yes"
## RAINFALL                    "No"
## GROUNDWATER                 "No"
## GEOTHERMAL                  "No"
## WATERQUALITY                "Yes"
## AIRQUALITY                  "No"
## LAND                        "No"
## CONSENT                     "No"
## EASTING                     "1950804"
## NORTHING                    "5786077"
## WEBPORTAL_SYNC              "true"
## LOCATIONTYPENAME            "Environmental Monitoring"
## ATTRIBUTETABLENAME          "Env_Mon_Extension"
## LOCATIONPATH                "All Locations.Whakatane and Tauranga.Whakatane"
## MONITORINGAGENCY            "NIWA/BOPRC"
## SURFACEWATERSTATIONID       "15514"
## RECNZREACHID                "4008243"
## RAINFALLSTATIONNETWORKID NA
## RAINFALLMETSERVICEID        NA
## GROUNDWELLID                NA
## GEOTHERMALFEATUREID         NA
## WATERQUALITYID              "BOP110098"
## AIRQUALITYID                NA
## LANDID                      NA
## CONSENT_NUMBER              NA
## MONITORING_CODE             NA
```

```r
# store the 'siteID' as an object that we can call
# upon later
SiteID <- WHKsites[31, 6]
SiteID
```

```
## [1] LL080607
## 51 Levels: JE727868 KE090494 KE107512 KE293801 KF432013 ... WMA 6
```

Great, now we know that the siteID is 'LL080607' and this value is stored as the object 'SiteID'. Now we can move on to the next step.

Discovering Data

Let's use another function 'datasets()' to see what data has been collected at the Valley Road site.

```
WHKVRData <- datasets(SiteID)
head(WHKVRData, 5)
```

```
##         Parameter LocationId                                DataId Unit.x
## 1              HG   LL080607              HG.Field Visits@LL080607      m
## 2              HG   LL080607                    HG.Master@LL080607      m
## 3              HG   LL080607                  HG.Hydrotel@LL080607      m
## 4        HGchange   LL080607        HGchange.Field Visits@LL080607     mm
## 5 HydraulicRadius   LL080607 HydraulicRadius.Field Visits@LL080607      m
##      Min         Max        Mean                     StartTime StartValue
## 1  0.075    7.306000   1.8027941 1952-07-10T13:30:00.000+12:00      1.201
## 2  0.053    8.354019   0.8379334 1956-07-31T15:00:00.000+12:00      1.051
## 3  0.000 8354.019165 897.3463731 2017-01-01T14:00:00.000+12:00    458.507
## 4 -0.366    0.000000  -0.0690000 1995-09-07T15:38:00.000+12:00     -0.037
## 5  0.487    3.706000   2.0748571 1995-09-07T15:38:00.000+12:00      3.169
##                        EndTime  EndValue TotalGaps TotalSamples
## 1 2017-12-05T12:20:00.000+12:00 0.4100000         0          505
## 2 2017-05-06T22:00:00.000+12:00 0.8447698       567       974220
## 3 2018-03-01T14:00:00.000+12:00 0.6286199         0       122107
## 4 1999-02-23T06:45:00.000+12:00 0.0000000         0            7
## 5 1999-02-23T06:45:00.000+12:00 0.4870000         0            7
##                   LastModified        DisplayId
## 1 2018-02-23T09:15:10.620+12:00            Stage
## 2 2017-12-19T11:27:23.017+12:00            Stage
## 3 2018-03-01T15:03:27.090+12:00            Stage
## 4 2018-02-23T09:15:10.223+12:00     Stage Change
## 5 2018-02-23T09:15:09.783+12:00 Hydraulic Radius
##                                 Name Interpolation Unit.y
## 1                        River Level             1      m
## 2                        River Level             1      m
## 3                        River Level             1      m
## 4 Stage Change During Location Visit             7     mm
## 5                   Hydraulic Radius             7      m
```

Again, you can either execute this code or click on WHKVRData in the Data pane.

```
View(WHKVRData)
```

The required input for the next function is called the 'DataID' (column 3 in WHKVRData). We are interested in flow, which is called 'QR' in Aquarius. In this case there are many flow derivatives but we are only interested in the raw data 'QR.Master'. To make things even more confusing, raw flow data at NIWA sites are called 'QR.Master' whereas the same data at BOPRC sites is called 'QR.Primary'. You may also notice that 'DataID' is simply a combination of the type of data and the site name, i.e. 'QR.Master@LL080607'.

The ‘getdata’ Function

The ‘getdata’ function forms the foundation for Aquarius data extraction within R. You will be introduced to another data extraction method later on called ‘AQMultiExtract’, but that function simply loops through multiple site and parameter combinations using the getdata function and then stitches everything together.

Take a look at column 13 in WHKVRData and you will notice that there are 973000+ records for ‘QR.Master’. It would take a long time to process if we extracted all that data. Luckily, the ‘getdata’ function that we use to call data allows us to specify a time frame in the format ‘YYYY-MM-DD’.

Let’s extract flow data from this site for the calendar year of 2014. Note that all inputs into the getdata function need to be character strings.

```
Flow <- getdata("QR.Master@LL080607", start = "2014-01-01",
    end = "2014-12-31")

head(Flow, 5)
```

```
##   RangeNumber                Time    Value Quality Interpolation Approval
## 1           1 2014-01-01 00:00:00 40.00000     200             1       60
## 2           1 2014-01-01 00:15:00 39.83855     200             1       60
## 3           1 2014-01-01 00:30:00 39.67748     200             1       60
## 4           1 2014-01-01 00:45:00 39.51677     200             1       60
## 5           1 2014-01-01 01:00:00 39.43656     200             1       60
```

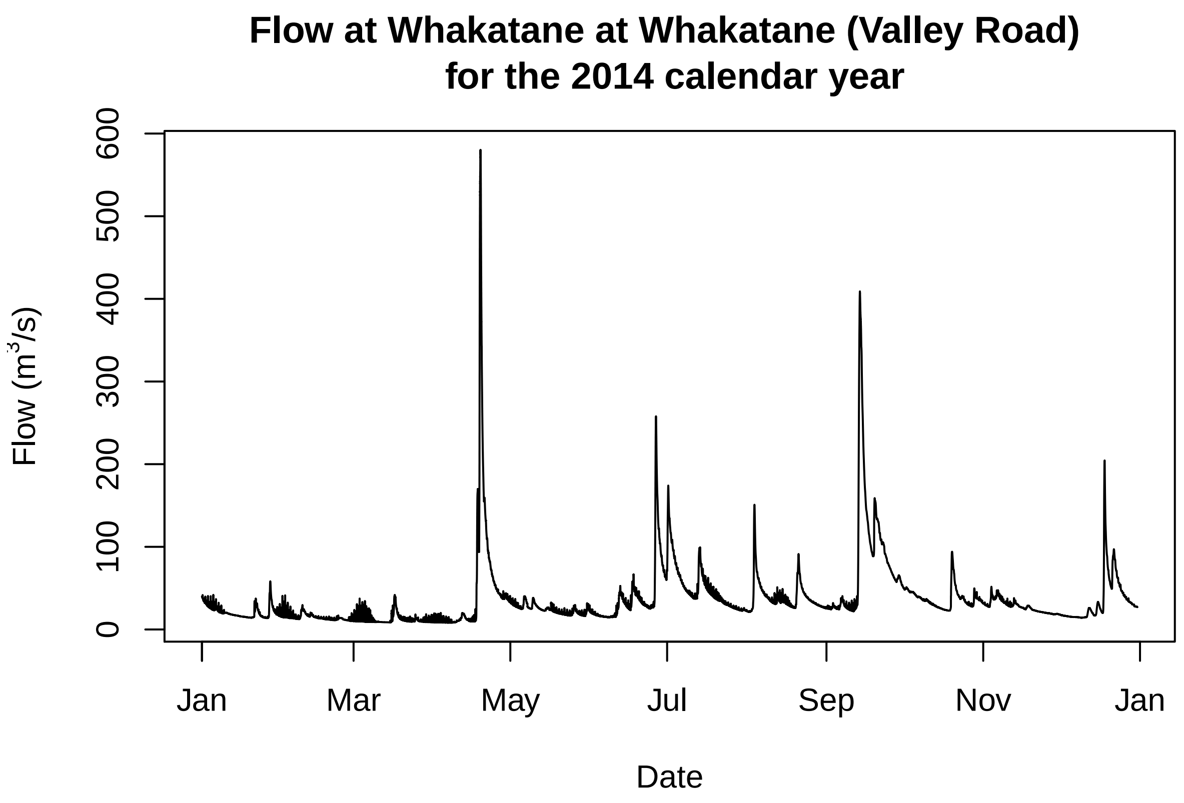
Plotting Flow Data

As you can see, we now have a dataframe with flow from the ‘Whakatane at Whakatane (Valley Road)’ site, for the year of 2014. Let’s plot this data, just because we can!

You may remember from Martin’s course that there a multiple plotting packages available. The ‘base’ plotting package is loaded when you start R and provides a ‘no-frills’ approach to plotting. When I say no-frills, you can definitely make your plots attractive but you may need more code than other packages.

```
# This code uses the base package to create an x-y plot

plot(x = Flow$Time,y=Flow$Value, type='l',
     main = paste("Flow at Whakatane at Whakatane (Valley Road)",
                  "\n","for the 2014 calendar year"),
     xlab = "Date",ylab = expression(paste("Flow (m"^3,"/s)",sep="")))
```



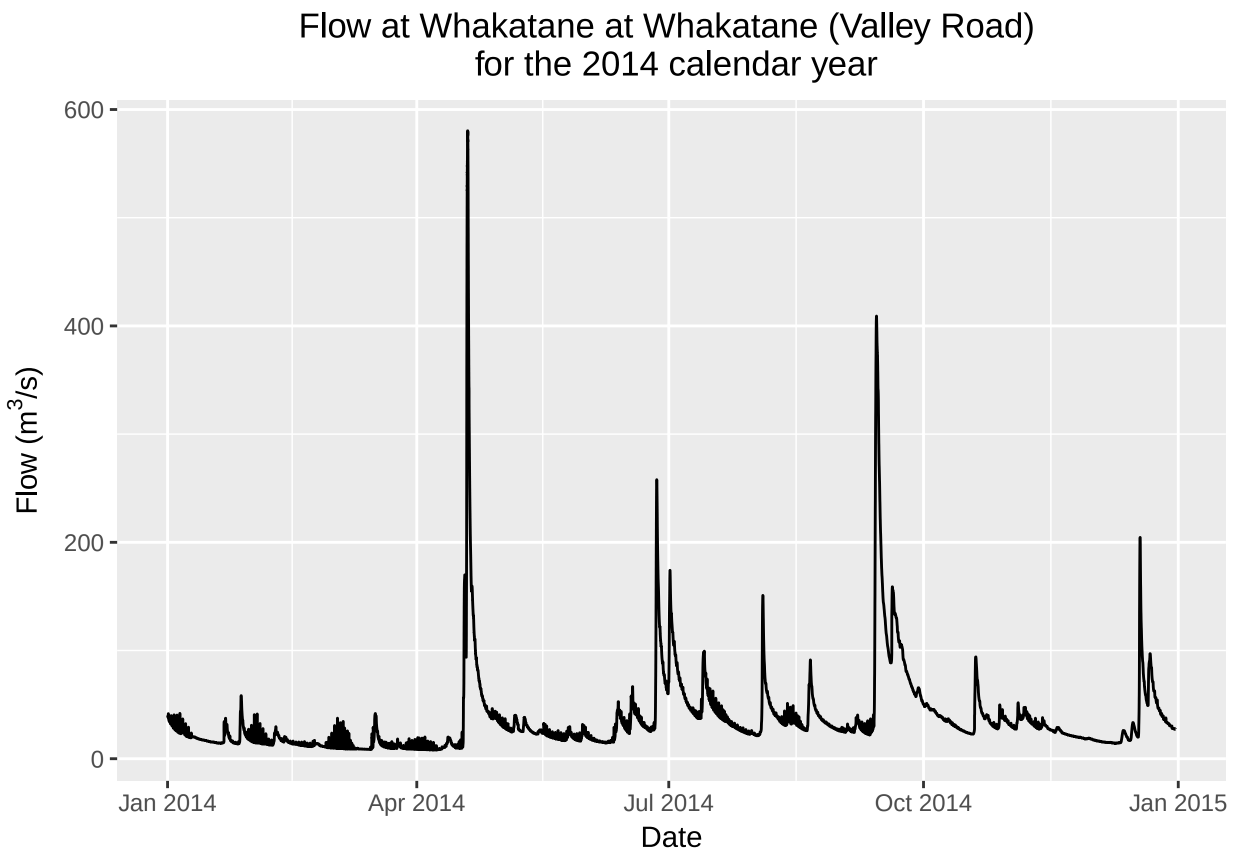
Another alternative is ggplot which, in the words of the developer, "takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics".

I find ggplot to be one of the more user-friendly plotting packages, with lots of support and a multitude of plotting options. You can find out more here: http://ggplot2.org/.

```
# load the ggplot2 package
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.2
```

```
# plot the flow data
ggplot(data = Flow, mapping = aes(x = Time, y = Value)) +
    geom_line() + labs(x = "Date", y = expression(paste("Flow (m"^3,
    "/s)", sep = ""))) + ggtitle(paste("Flow at Whakatane at Whakatane (Valley Road)",
    "\n", "for the 2014 calendar year")) + theme(plot.title = element_text(hjust = 0.5))
```



The choice is yours in the end, but I recommend learning one package well using this for the majority of your day-to-day plots.

Extracting Water Quality Data

Great, we have flow for the Whakatane River, but what about water quality metrics? Sometimes these will be available at the same site as the flow record, but often water quality and flow are measured separately and have to be linked together.

For the next part of this tutorial we will extract water quality data from the Whakatane at Pekatahi Bridge site, and then append each sample with flow from the continuous flow record (Whakatane at Whakatane [Valley Road]).

First, lets go back to the dataframe 'WHKsites' to find 'Whakatane at Pekatahi Bridge'

```r
# Whakatane at Pekatahi Brige is row 13 in the
# WHKsites dataframe
t(WHKsites[13, ])
```

```
##                              13
## LOCATIONID                   "16414"
## LOCATIONFOLDERID             "89"
## LASTMODIFIED                 "2016-08-22T16:02:30.447"
## LOCATIONNAME                 "Whakatane at Pekatahi Bridge"
## DESCRIPTION                  NA
## IDENTIFIER                   "KL998150"
## LOCATIONTYPEID               "1000"
## LATITUDE                   "-38.04655"
## LONGITUDE                  "176.988"
## SRID                       "4326"
## ELEVATIONUNITS             "m"
## ELEVATION                  "6"
## UTCOFFSET                  "12"
## TIMEZONE                   NA
## DEFAULTROLEID              NA
## AQUSERDATA_                NA
## REGIONALMONITORINGSITE     "Current"
## AUTOMATED                  "No"
## SURFACEWATERLEVEL          "No"
## SURFACEWATERFLOW           "Yes"
## RAINFALL                   "No"
## GROUNDWATER                "No"
## GEOTHERMAL                 "No"
## WATERQUALITY               "Yes"
## AIRQUALITY                 "No"
## LAND                       "No"
## CONSENT                    "No"
## EASTING                    "1949980"
## NORTHING                   "5781507"
## WEBPORTAL_SYNC             "true"
## LOCATIONTYPENAME           "Environmental Monitoring"
## ATTRIBUTETABLENAME         "Env_Mon_Extension"
## LOCATIONPATH               "All Locations.Whakatane and Tauranga.Whakatane"
## MONITORINGAGENCY           "BOPRC"
## SURFACEWATERSTATIONID      "15543"
## RECNZREACHID               "4010027"
## RAINFALLSTATIONNETWORKID NA
## RAINFALLMETSERVICEID       NA
## GROUNDWELLID               NA
## GEOTHERMALFEATUREID        NA
## WATERQUALITYID             "BOP110011"
## AIRQUALITYID               NA
## LANDID                     NA
## CONSENT_NUMBER             NA
## MONITORING_CODE            NA
```

```
# Save the site ID as object 'SiteID2'
SiteID2 <- WHKsites[13, 6]
SiteID2
```

```
## [1] KL998150
## 51 Levels: JE727868 KE090494 KE107512 KE293801 KF432013 ... WMA 6
```

The function we are going to use requires a list of sites and a list of parameters as input. Our list of sites has been created above (SiteID2) so now we need to create a list of parameters.

The 'LocationWQParameters' Function

The 'LocationWQParameters' function provides the display names of all discrete water quality parameters available at a given site. The display name is the name that you see in 'Aquarius Springboard' and is the required input for the AQMultiExtract function, e.g. TN = "N (Tot)"

We input the SiteID2 object above to provide us with this list of water quality parameters available at the Pekatahi Bridge site.

```r
# Create an object with all discrete WQ parameters
# at the site.
AvailableWQParams <- LocationWQParameters(SiteID2)
AvailableWQParams
```

```
##  [1] "Conductivity"           "Fluoride"
##  [3] "Ammoniacal N"           "Nitrate (N)"
##  [5] "Nitrite (as N)"         "Sulphate"
##  [7] "TKN"                    "N (Tot)"
##  [9] "Nitrite Nitrate (as N)" "Faecal coliforms"
## [11] "Enterococci"            "Alkalinity"
## [13] "B (Dis)"                "BOD"
## [15] "Ca (Dis)"               "Cations"
## [17] "Cl (Dis)"               "Water Clarity"
## [19] "DRP"                    "Silica (Dis)"
## [21] "E coli"                 "Conductivity (Field)"
## [23] "Salinity (Field)"       "HCO3"
## [25] "Lab Stage"              "K (Dis)"
## [27] "Mg (Dis)"               "Na (Dis)"
## [29] "P (Tot)"                "pH"
## [31] "Water Temp"             "254nm UV Abs"
## [33] "340nm UV Abs"           "440nm UV Abs"
## [35] "740nm UV Abs"           "Tot Susp Sed"
## [37] "O2 (Dis)"               "Turbidity"
## [39] "Dis Oxygen Sat"
```

We may not be interested in all of the available data, so let's create a subset that specifies: NNN, TN, NH4 + NH3, TP, DRP, E. Coli, and pH. To do this, we could either write out the display names using 'c("N (Tot)","Nitrite Nitrate (as N)", . . .)' and assign it to an object, or we could refer to the object we created above and use the row numbers of the parameters we are interested in.

```r
ParamList <- AvailableWQParams[c(9, 8, 3, 29, 19, 21,
    30)]
ParamList
```

```
## [1] "Nitrite Nitrate (as N)" "N (Tot)"
## [3] "Ammoniacal N"           "P (Tot)"
## [5] "DRP"                    "E coli"
## [7] "pH"
```

The AQMultiExtract Function

This function is the most useful tool in the Aquarius package toolbox, for extracting large datasets with multiple sites and multiple parameters. You can also use it for extracting a single site with multiple parameters (as in this case), or multiple sites with a single parameter.

The syntax requirements for AQMultiExtract are:

- **sitelist** - a list of site ID's.
- **params** - a list of parameters as display names.
- **start** - (optional) start date in the format YYYY-MM-DD.
- **end** - (optional) end date in the format YYYY-MM-DD.

In this case the sitelist is a list of one, i.e. SiteID2, and the paramlist is that above. We will analyse data between the 1st January 2007 and the 1st January 2017 (i.e. a ten year period).

```
WQData_Pekatahi <- AQMultiExtract(SiteID2, ParamList,
    start = "2007-01-01", end = "2017-01-01")
```

```
## .......
```

```
head(WQData_Pekatahi)
```

```
##       Site                 LocationName                Time
## 1 KL998150 Whakatane at Pekatahi Bridge 2007-01-22 12:20:00
## 2 KL998150 Whakatane at Pekatahi Bridge 2007-02-13 13:55:00
## 3 KL998150 Whakatane at Pekatahi Bridge 2007-03-14 13:50:00
## 4 KL998150 Whakatane at Pekatahi Bridge 2007-04-17 12:40:00
## 5 KL998150 Whakatane at Pekatahi Bridge 2007-05-03 12:59:00
## 6 KL998150 Whakatane at Pekatahi Bridge 2007-06-12 13:45:00
##   Ammoniacal N (g/m^3) DRP (g/m^3) E coli (cfu/100ml) N (Tot) (g/m^3)
## 1               0.0030       0.021                210              NA
## 2               0.0030       0.021                 82              NA
## 3               0.0220       0.034               2000              NA
## 4               0.0050       0.015                160              NA
## 5               0.0040       0.012                510              NA
## 6               0.0005       0.006                150              NA
##   Nitrite Nitrate (as N) (g/m^3) P (Tot) (g/m^3) pH (pH Units)
## 1                         0.0190           0.049          7.20
## 2                         0.0090           0.033          7.72
## 3                         0.1300              NA          7.40
## 4                         0.0160           0.024          7.60
## 5                         0.0005           0.027          7.20
## 6                         0.0005           0.027          7.70
```

It's as easy as that. You could save this to a csv file using the code below if you wish, but we will carry on.

```
write.csv(WQData_Pekatahi, file = "FILEPATH.csv", row.names = FALSE)
```

Appending Flow to Water Quality Data

In this section we will append mean daily flow values from a continuous flow record to the water quality dataset we created above. This tool can also be used to match up spot gauging values with water quality samples, as the timestamps don't usually match up, therefore they can't be extracted together via AQMultiExtract.

The first step is to extract a ten year dataset to match our water quality dataset.

We can use the same methodology as before to extact a 10 year flow record from this site. This may take a while as 10 years of continuous data is a lot!

```
Flow10YR <- getdata("QR.Master@LL080607", start = "2007-01-01",
    end = "2017-01-01")
```

The AQAppendFlow Function

Now that we have a WQ dataset and a continuous flow record, we want to match time stamps to determine the flow when samples were taken. We could either find out the timestamp for each sample and then trawl through the flow record to determine the best match, or we could use our newly developed R skills…

To assist in this task, we have developed a function called 'AQAppendFlow'. The purpose of this is to append flow values from another source (dataframe) to a separate dataframe containing discrete water quality information. The function works by looking at the timestamp for each sample in your WQ dataset, and then calculating the difference between that and each data line in the flow record. You also need to specify the time bracket that flow timestamps can precede or proceed the WQ timestamp (e.g. +- 5 days). If there are more than one available flow values within the specified bracket, the function will take the value that is closet to the WQ timestamp.

You can use AQAppendFlow for linking WQ data to discrete (spot gauging) or continuous flow. Note that spot gauging data will have a different timestamp to WQ samples and therefore doesn't line up when using AQMultiExtract.

For the current task, we have a dataset of continuous (15 min) flow data that we want to append to the dataset 'WQData_Pekatahi'. It also makes more sense to use mean daily flow, as matching WQ time stamps to the nearest 15 minute flow stamp does not seem like an appropriate representation of the general state of the river when the sample was taken. This would also require a great deal more processing!

Before we start the process, we need to turn our continuous flow record into a mean daily record (note that some Aquarius sites have a pre-calculated mean daily flow record, and it may be easier just to extract that). We can do this using the 'FlowtoMeanDaily' function. This only requires a continuous flow record to run.

```
Flow10YR_MD <- FlowtoMeanDaily(Flow10YR)
head(Flow10YR_MD)
```

```
##   RangeNumber        Day     Flow
## 1           1 2007-01-01 31.23191
## 2           1 2007-01-02 29.56367
## 3           1 2007-01-03 23.56466
## 4           1 2007-01-04 21.69418
## 5           1 2007-01-05 20.14301
## 6           1 2007-01-06 19.19713
```

Note that the observations reduced from 456813 to 3654. This is much more manageable for our analysis.

The AQAppendFlow function allows multiple flow sites to be appended to multiple WQ sites by matching up site IDs. We only have one WQ site in this case, but our flow data is lacking the required information to tell the function which site it belongs to. We can do this by modifying the flow dataframe so it matches the required input: ("ID","Name","Time", "Discharge").

```
# we are changing the ID so it can be matched to
# the WQ dataset
Flow10YR_MD$ID <- "KL998150"

# the same goes for the name
Flow10YR_MD$Name <- "Whakatane at Pekatahi Bridge"

# final flow dataset
Flow10YR_MD <- Flow10YR_MD[, c(4, 5, 2, 3)]

head(Flow10YR_MD)
```

```
##         ID                         Name        Day     Flow
## 1 KL998150 Whakatane at Pekatahi Bridge 2007-01-01 31.23191
## 2 KL998150 Whakatane at Pekatahi Bridge 2007-01-02 29.56367
## 3 KL998150 Whakatane at Pekatahi Bridge 2007-01-03 23.56466
## 4 KL998150 Whakatane at Pekatahi Bridge 2007-01-04 21.69418
## 5 KL998150 Whakatane at Pekatahi Bridge 2007-01-05 20.14301
## 6 KL998150 Whakatane at Pekatahi Bridge 2007-01-06 19.19713
```

Now we can append flow to the WQ dataset using the AQAppendFlow function. This requires: a dataset of WQ data (i.e. WQData_Pekatahi), a dataset of flow values (i.e. Flow10YR_MD), and a bracket of acceptable duration (in days). Lets use 1 day as this is a continuous record and there should be data within one day of the WQ timestamp.

```r
WQData_Appended <- AQAppendFlow(WQData_Pekatahi, Flow10YR_MD,
    1)
head(WQData_Appended)
```

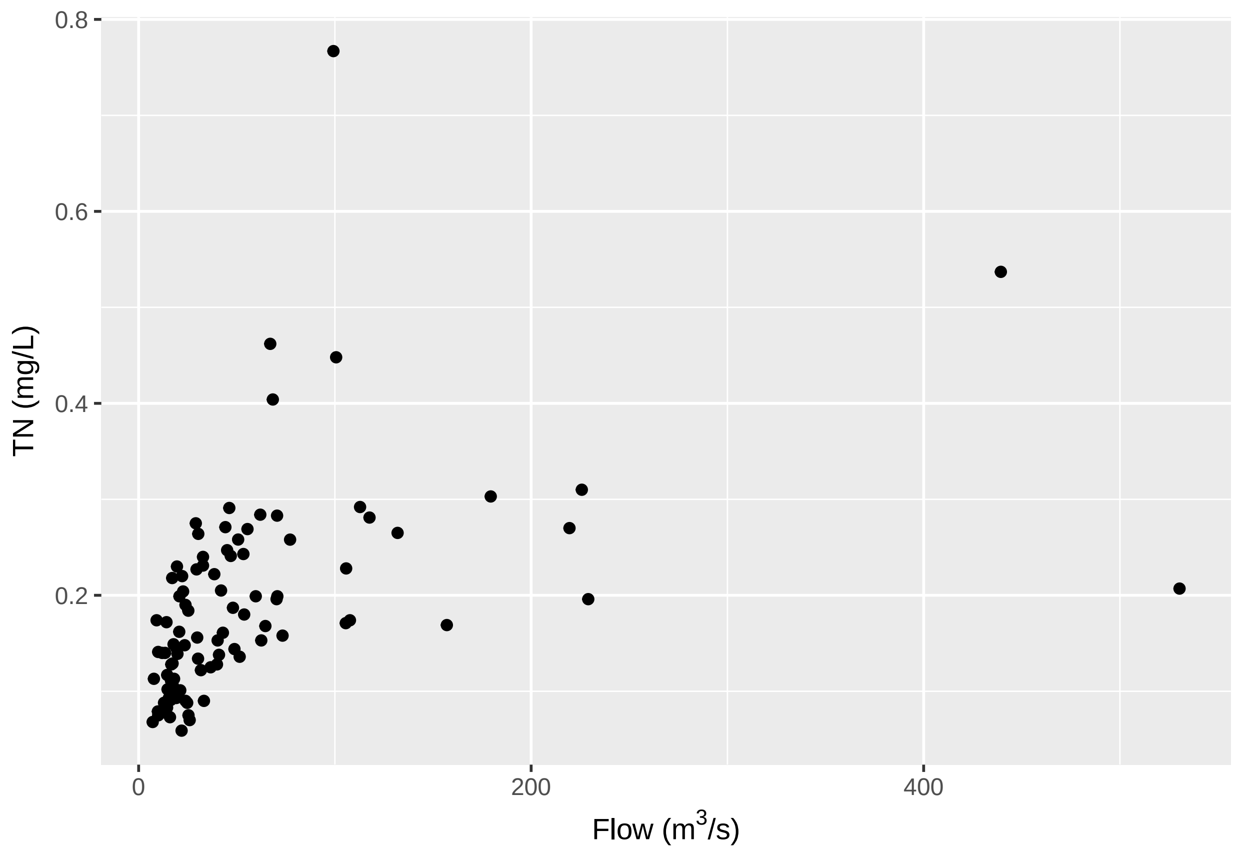
```
##       Site                 LocationName                Time
## 1 KL998150 Whakatane at Pekatahi Bridge 2007-01-22 12:20:00
## 2 KL998150 Whakatane at Pekatahi Bridge 2007-02-13 13:55:00
## 3 KL998150 Whakatane at Pekatahi Bridge 2007-03-14 13:50:00
## 4 KL998150 Whakatane at Pekatahi Bridge 2007-04-17 12:40:00
## 5 KL998150 Whakatane at Pekatahi Bridge 2007-05-03 12:59:00
## 6 KL998150 Whakatane at Pekatahi Bridge 2007-06-12 13:45:00
##   Ammoniacal N (g/m^3) DRP (g/m^3) E coli (cfu/100ml) N (Tot) (g/m^3)
## 1               0.0030       0.021                210              NA
## 2               0.0030       0.021                 82              NA
## 3               0.0220       0.034               2000              NA
## 4               0.0050       0.015                160              NA
## 5               0.0040       0.012                510              NA
## 6               0.0005       0.006                150              NA
##   Nitrite Nitrate (as N) (g/m^3) P (Tot) (g/m^3) pH (pH Units)     Flow
## 1                         0.0190           0.049          7.20 47.96759
## 2                         0.0090           0.033          7.72 21.01343
## 3                         0.1300              NA          7.40 36.91703
## 4                         0.0160           0.024          7.60 18.96507
## 5                         0.0005           0.027          7.20 40.21187
## 6                         0.0005           0.027          7.70 21.99773
```

Success! We now have a WQ dataset from Pekitahi Bridge that is appended with flow from the Valley Road continuous monitoring site. Note that this is just an example and there may be many reasons why you would not do this (talk to EDS before attempting).

We can now use our plotting skills to look at how individual parameters vary with flow.

```r
# change the names into something more useful
names(WQData_Appended) <- c("SiteID", "Name", "Time",
    "NH4", "DRP", "ECOLI", "TN", "NNN", "TP", "pH",
    "Flow")

# plot the data
ggplot(WQData_Appended, aes(x = Flow, y = TN)) + geom_point(na.rm = TRUE) +
    ylab("TN (mg/L)") + xlab(expression(paste("Flow (m"^3,
    "/s)", sep = "")))
```



Perhaps we would like to create ‘flow bins’ based on flow quantiles, and then see how TN varies.

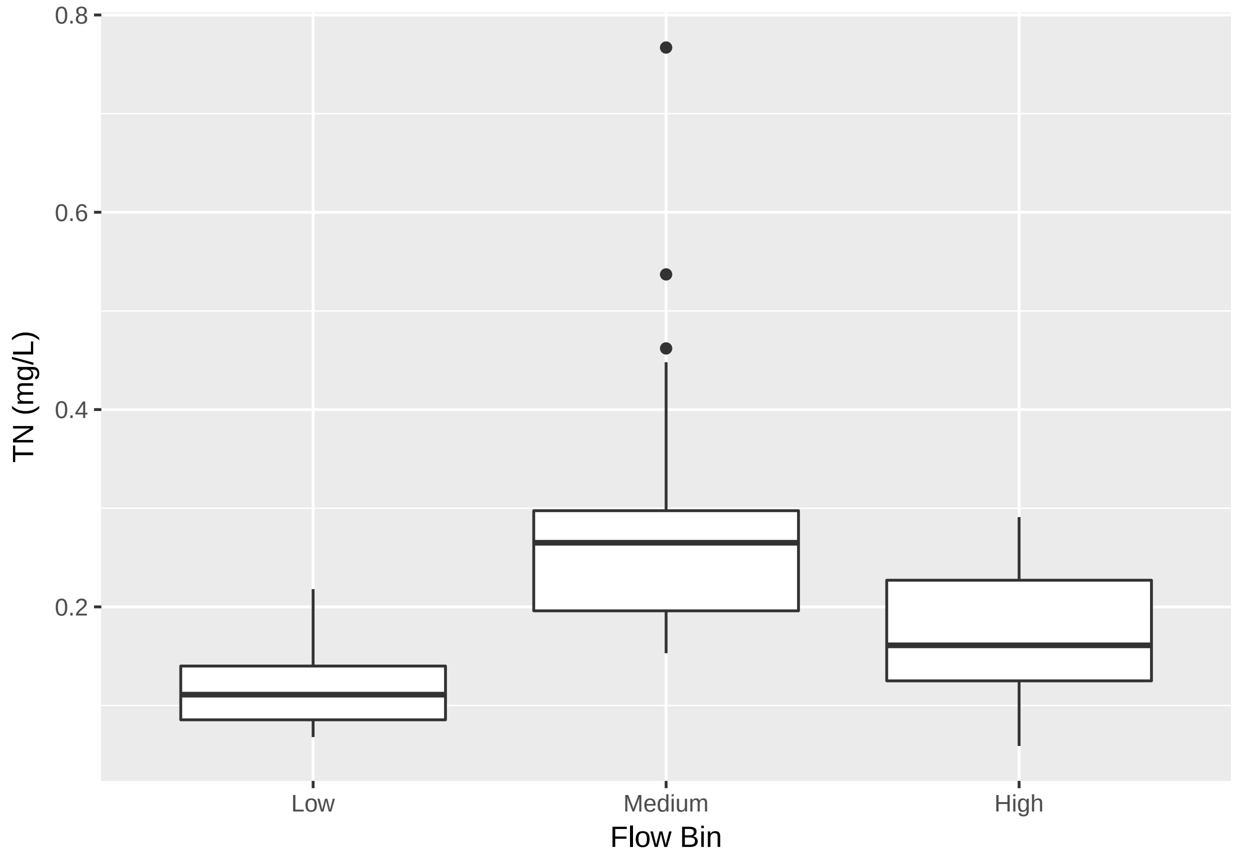
```r
# calculate flow quantiles
quantile(WQData_Appended$Flow)
```

```
##         0%        25%        50%        75%       100%
##   7.152449  17.992135  30.325067  54.199848 530.379879
```

```r
# use flow quantiles to specify factor levels
WQData_Appended$Flowbin <- ifelse(WQData_Appended$Flow <
    17.992135, "Low", ifelse(WQData_Appended$Flow >=
    54.199848, "Medium", "High"))

# reorder the factor levels so they make sense (the
# deafault is alphabetical)
WQData_Appended$Flowbin <- as.factor(WQData_Appended$Flowbin)
WQData_Appended$Flowbin <- factor(WQData_Appended$Flowbin,
    levels = c("Low", "Medium", "High"))

# create a boxplot
ggplot(WQData_Appended, aes(x = Flowbin, y = TN)) +
    geom_boxplot(na.rm = TRUE) + ylab("TN (mg/L)") +
    xlab("Flow Bin")
```



The BoPRC Library

The BoPRC library is intended as a suite of useful analyses and data manipulation methods to make life a bit simpler for BoPRC R users. The intention is to expand this suite as users write or discover functions that may be useful for other staff. This is a great way to achieve standardised and auditable outputs for reporting purposes. The options for future development are unlimited, so make sure you discuss any ideas you have you have with James.

To get started, let's load the BoPRC package so we can access these tools.

```r
library(BoPRC)
```

NOF Reporting

Reporting against the NOF can be a mundane exercise, and many attributes are open to interpretation. For this reason we have created a function to process large datasets using standardised methodology. In this example we will demonstrate the two most complicated functions: NOFLakesRiversNH3, and NOFLakesRiversECOLI. However, there are also functions to assess against other NOF tables. Details of these are listed at the bottom of this section under 'List of NOF Functions', but you can also search the BoPRC help file for more information.

First let's create a dummy river water quality dataset using the Aquarius package.

```r
# random list of sites taken from the SOE network
Sites <- c("BQ966369", "EL204387", "EL613536", "RN123610",
    "FN834668", "FL356693", "FO620177", "RO629568",
    "DP784306", "DO406909", "GI416337", "QM756918",
    "QJ471191", "CO543022", "EL174017", "MK788545")

# parameters required for this analysis
Parameters <- c("E coli", "Ammoniacal N", "pH")

# use AQMultiExtract to pull out data from multiple
# site/parameter combinations.
RiverWQ <- AQMultiExtract(Sites, Parameters, start = "2010-01-01",
    end = "2017-01-01")
```

```
## ................................................
```

```
head(RiverWQ)
```

```
##       Site     LocationName                Time Ammoniacal N (g/m^3)
## 1 BQ966369 Aongatete at SH2 2010-07-13 14:05:00                0.008
## 2 BQ966369 Aongatete at SH2 2010-08-23 14:15:00                0.005
## 3 BQ966369 Aongatete at SH2 2010-09-07 14:00:00                0.009
## 4 BQ966369 Aongatete at SH2 2010-10-19 13:27:00                0.007
## 5 BQ966369 Aongatete at SH2 2010-11-18 13:00:00                0.008
## 6 BQ966369 Aongatete at SH2 2010-12-02 12:38:00                0.030
##   E coli (cfu/100ml) pH (pH Units)
## 1                  4           7.0
## 2                 13           7.1
## 3                130           7.4
## 4                 19           7.2
## 5                 34           7.2
## 6                430           7.5
```

Ammonia Toxicity

Let’s start with Ammonia Toxicity (NH3-N calculated from NH4-N and pH).

The function ‘NOFLakesRiversNH3’ requires the following inputs:

- **data** -user to input a dataframe in the format: SiteID, Site Name, Time (in YYYY-DD-MM), Value, pH.

- **start** - (optional) in the format YYYY-MM-DD.
- **end** - (optional) in the format YYYY-MM-DD.
- **adjust** (optional) a boolean adjustment parameter (either TRUE or FALSE). If TRUE, the NH4-N values will be adjusted for pH, assuming a temperature of 20.0 degrees. The default is FALSE.

```
# subset the RiverWQ data so it is in the correct
# format
RiverWQNH3 <- RiverWQ[c(1, 2, 3, 4, 6)]

# get rid of rows with empty values
RiverWQNH3 <- RiverWQNH3[complete.cases(RiverWQNH3),
    ]

# run summary function
NH3Summary <- NOFLakesRiversNH3(RiverWQNH3, start = "",
    end = "", adjust = TRUE)
```

```
## [1] "Data adjusted to pH 8.0 and temperature of 20deg"
```

```r
head(NH3Summary, 14)
```

```
##                                Name      Minimum     Maximum      Median  n
## 1                  Aongatete at SH2 0.0004310345 0.019018404 0.002066116 43
## 2                    Awahou at SH36 0.0003663004 0.059073359 0.001465202 79
## 3           Hamurana at Hamurana Rd 0.0004132231 0.017160000 0.002272727 73
## 4                Haparapara at SH35 0.0000000000 0.024621211 0.001074123 42
## 5         Kaituna at Maungarangi Rd 0.0003571429 0.017045455 0.002990102 82
## 6         Kaituna at Rotoiti Outlet 0.0000000000 0.037180000 0.006047545 84
## 7               Kaituna at Te Matai 0.0028925621 0.174725275 0.014444445 66
## 8                     Kereu at SH35 0.0000000000 0.006198347 0.001147332 42
## 9                Kopurererua at SH2 0.0008264463 0.166795367 0.021621622 81
## 10              Kopurererua at SH29 0.0000000000 0.050431034 0.005429864 83
## 11 Mangakino at Rerewhakaaitu Road 0.0004784689 0.312500009 0.009561753 85
## 12        Ngamuwahine at Old Bridge 0.0003861004 0.005791506 0.002110644 42
## 13               Ngongotaha at SH36 0.0020661157 0.037180000 0.006334842 83
## 14          Nukuhou at Glenholme Rd 0.0028708134 0.052123552 0.013944223 27
##    AttributeBand
## 1              A
## 2              B
## 3              A
## 4              A
## 5              A
## 6              A
## 7              B
## 8              A
## 9              B
## 10             B
## 11             B
## 12             A
## 13             A
## 14             B
```

Success! You can easily export this table to Excel using the 'Write.Excel' function. I find this is the easiest way to manipulate tables for reports or presentations.

```r
Write.Excel(NH3Summary)
```

Swimmability

Assessing swimmability against the NOF used to be a straightforward exercise, but now there are multiple attributes to consider when calculating a NOF band, and sometimes the logic doesn't quite stack up. The 'NOFLakesRiversECOLI' function provides a tool with which swimmability can be assessed, but requires the user to make up their own mind when things don't work (i.e. bands do not agree between numeric attributes).

The function requires the following inputs:

- **data** - in the usual format of: SiteID, Site Name, Time (in YYYY-DD-MM), Value.

- **start** - (optional) in the format YYYY-MM-DD
- **end** - (optional) in the format YYYY-MM-DD.

```r
# subset the RiverWQ data so it is in the correct
# format
RiverECOLI <- RiverWQ[c(1, 2, 3, 5)]
```

```r
# get rid of rows with empty values
RiverECOLI <- RiverECOLI[complete.cases(RiverECOLI),
    ]

# run summary function
NOFLakesRiversECOLI(RiverECOLI)
```

```
##                                Name Minimum    Maximum Median
## 1                   Aongatete at SH2       3  3100.0000   41.0
## 2            Hamurana at Hamurana Rd      22   113.3333   46.0
## 3                 Haparapara at SH35       0  3000.0000    4.5
## 4           Kaituna at Maungarangi Rd       2   300.0000   24.0
## 5           Kaituna at Rotoiti Outlet       0   210.0000    5.0
## 6                 Kaituna at Te Matai       4  3700.0000   45.0
## 7                      Kereu at SH35       0  7000.0000    5.5
## 8                 Kopurererua at SH2      21  4400.0000  165.0
## 9                Kopurererua at SH29      27  6800.0000  260.0
## 10 Mangakino at Rerewhakaaitu Road      14  7000.0000  180.0
## 11      Motu at Houpoto (NIWA site)       0  1200.0000   18.0
## 12 Motu at Waitangirua (NIWA site)       1  6000.0000   78.0
## 13         Ngamuwahine at Old Bridge       4 10000.0000   36.0
## 14                Ngongotaha at SH36       2  9666.6667  170.0
## 15           Nukuhou at Glenholme Rd      53 12800.0000  230.0
##    Exceedances 540 Exceedances 260  n Percentile95 PercentExceed540
## 1                3               6 44     614.0000         6.818182
## 2                0               0  9     102.6667         0.000000
## 3                2               4 84      83.4000         2.380952
## 4                0               2 81     110.0000         0.000000
## 5                0               0 84      25.8500         0.000000
## 6                2               5 66     295.0000         3.030303
## 7                3               3 88     120.6667         3.409091
## 8               11              24 82    2175.0000        13.414634
## 9               17              39 83    1790.0000        20.481928
## 10              17              31 83    1727.0000        20.481928
## 11               5               8 86     665.0000         5.813953
## 12              15              21 80    1640.0000        18.750000
## 13               6               9 54    1970.0000        11.111111
## 14              12              27 83    2083.0000        14.457831
## 15               9              11 27    3184.0000        33.333333
##    PercentExceed260 Primary Contact Secondary Contact ProposedBand
## 1         13.636364            Fail                 A        Green
## 2          0.000000               A                 A         Blue
## 3          4.761905               A                 A         Blue
## 4          2.469136               A                 A         Blue
## 5          0.000000               A                 A         Blue
## 6          7.575758               B                 A         Blue
## 7          3.409091               A                 A         Blue
## 8         29.268293            Fail                 A        Error
## 9         46.987952            Fail                 A       Orange
## 10        37.349398            Fail                 A       Orange
## 11         9.302326            Fail                 A        Green
## 12        26.250000            Fail                 A        Error
## 13        16.666667            Fail                 A        Error
## 14        32.530120            Fail                 A        Error
## 15        40.740741            Fail                 A        Error
##    ProposedSwimmable?
## 1           Swimmable
## 2           Swimmable
## 3           Swimmable
## 4           Swimmable
## 5           Swimmable
## 6           Swimmable
## 7           Swimmable
## 8               Error
## 9       Not Swimmable
## 10      Not Swimmable
## 11          Swimmable
## 12              Error
## 13              Error
## 14              Error
## 15              Error
```

The output provides a breakdown of the attributes used to calculate both the former NOF attribute bands (i.e. primary and secondary contact), and the new NOF bands. The function will also tell you if the site is swimmable under the respective guidelines.

In this case you can see that 'Kopurererua at SH2' throws an error for the new guidelines. This is because the four numeric attributes do not align with each other. In this case the percent exceedance attributes place the site in the 'yellow' band, the median places the site in the 'orange' band, and 95th percentile put the site in the 'red' band. The NOF states that attribute state must be determined by satisfying all numeric attribute states, therefore this is left up to the user to interpret.

List of NOF Functions

NOF functions exist for the following attributes:

- **Lakes**
  - Total Nitrogen (NOFLakesTN)
  - Total Phosphorus (NOFLakesTP)
  - Phytoplankton (NOFLakesPhytoplankton)
- **Rivers**
  - Periphyton (NOFRiversPeriphyton)
  - Nitrate (NOFRiversNO3)
- **Lakes and Rivers**
  - Ecoli (NOFLakesRiversECOLI)
  - Ammonia (NOFLakesRiversNH3)

They all have similar input requirements, please refer to the help files for more information.

Other Interesting Functions

This section looks at other functions in the BoPRC package that you may find useful. These are less important to the tutorial, so feel free to explore them at your own will.

LAWA Distribution Plots

This tool came about via discussion with Rebecca Lawton about the best way to communicate 'state' in absence of a useful framework. The idea is to compare the median of a site to a distribution that you wish to compare against (e.g. all like sites in the BoP, or all like sites in NZ), and then display the quartile group in an easy-to-read manner. Essentially this enables us to report in the same way as can be found on LAWA.

The first step in doing this is to load a dataset that will be used for comparison. In this case we will create one using the rnorm function.

```
National_TN <- rnorm(1000, mean = 0.23, sd = 0.09)
head(National_TN, 20)
```

```
##  [1] 0.2532502 0.1885207 0.3922502 0.1383894 0.2220871 0.2471232 0.3252945
##  [8] 0.2751706 0.1106767 0.2219664 0.3733773 0.1708228 0.1956450 0.1471501
## [15] 0.1502462 0.3653318 0.1612633 0.1958102 0.2519815 0.3069865
```

Next we can convert this into an create an 'Empirical Cumulative Distribution' using the ecdf. This function creates a cumulative distribution from the dataset we just created.

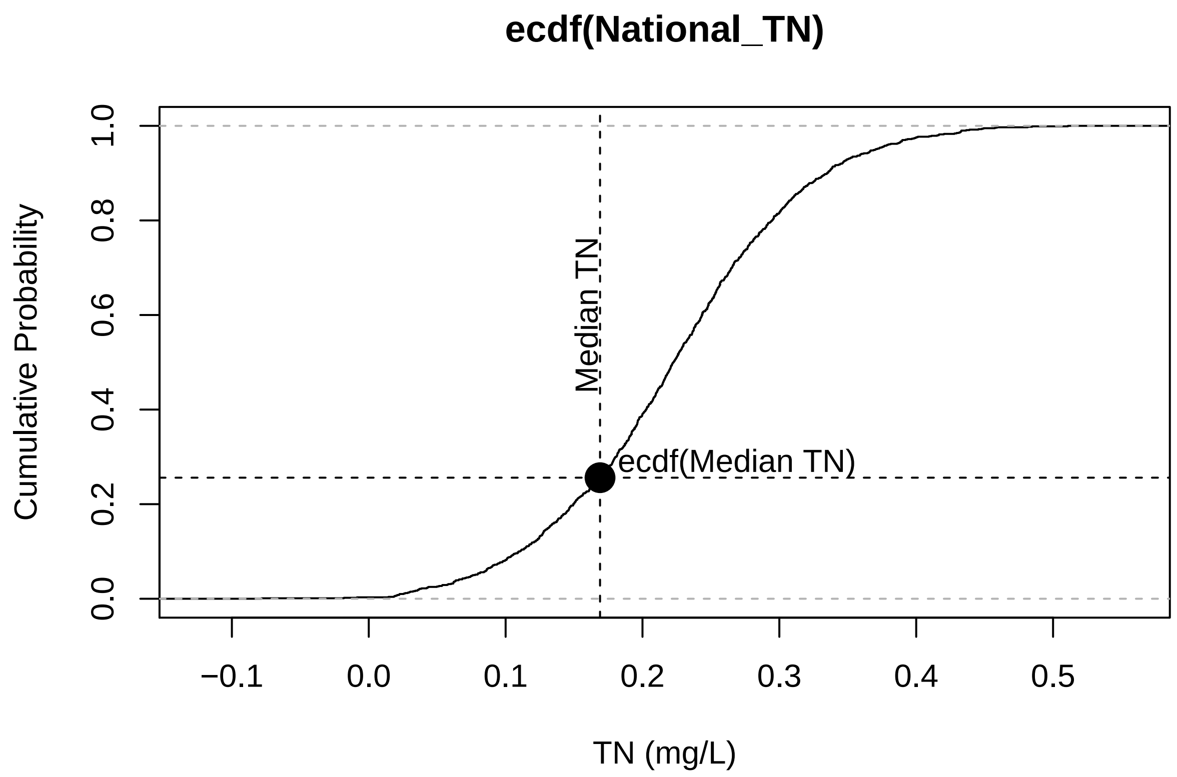
```
National_TN_ecdf <- ecdf(National_TN)
```

We can then query the distribution to determine where our comparison value sits, in this case we want to know how the median TN value from the WQData_Appended dataset compares to our national dataset.

```
National_TN_ecdf(median(WQData_Appended$TN, na.rm = T))
```

```
## [1] 0.256
```

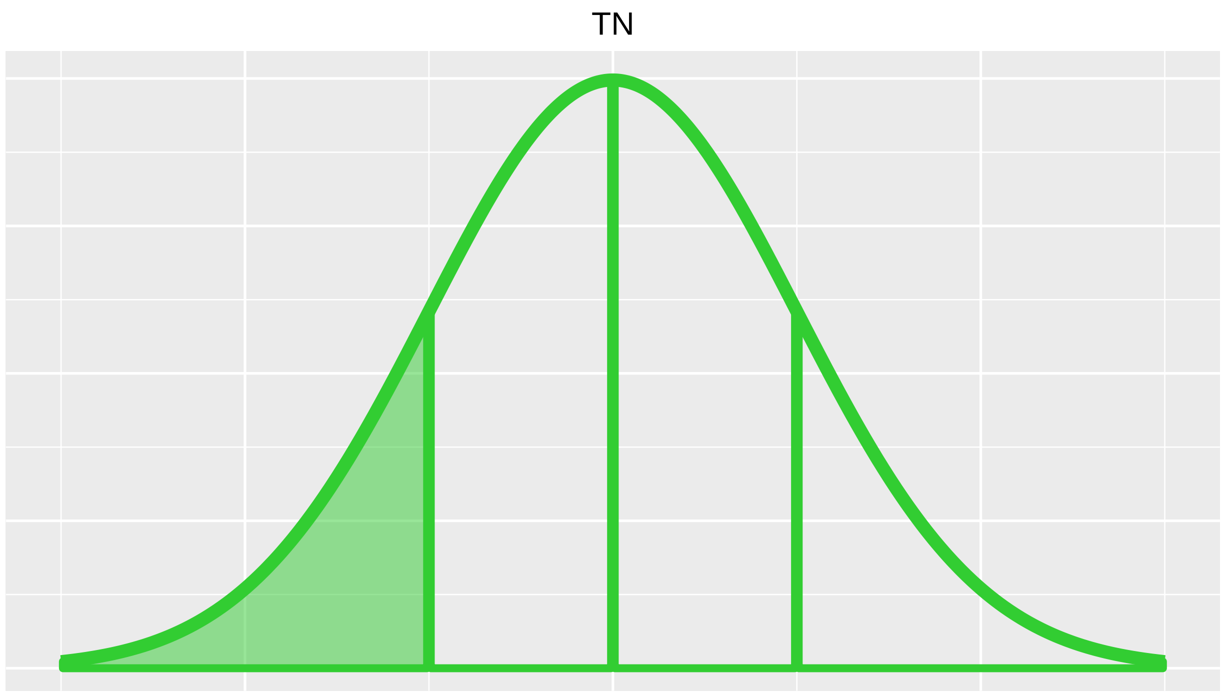
```
# example plot of the ecdf function
plot(National_TN_ecdf, xlab = "TN (mg/L)", ylab = "Cumulative Probability")
points(x = median(WQData_Appended$TN, na.rm = T), y = National_TN_ecdf(median(WQData_Appended$TN,
    na.rm = T)), pch = 19, cex = 2)
abline(h = National_TN_ecdf(median(WQData_Appended$TN,
    na.rm = T)), v = median(WQData_Appended$TN, na.rm = T),
    lty = 2)
text(x = median(WQData_Appended$TN, na.rm = T) + 0.1,
    y = National_TN_ecdf(median(WQData_Appended$TN,
        na.rm = T)) + 0.03, labels = "ecdf(Median TN)")
text(x = median(WQData_Appended$TN, na.rm = T) - 0.01,
    y = 0.6, labels = "Median TN", srt = 90)
```



This tells us that our median value falls in the 21st percentile of the national distribution. This can be confusing to the public, but luckily we can present the results in an easy to understand graphic using the LAWADistPlots function. This function creates a normal distribution and then fills in the area, and changes the colour, corresponding to a specified 'group' number. The group number refers to the percentile area that you want shaded: 1 = 0-25%, 2=25-50%, 3=50-75%, 4=75-100%. Note that this only works with parameters where higher values are perceived as being worse.

In the case above, the 21st percentile corresponds to group 1.

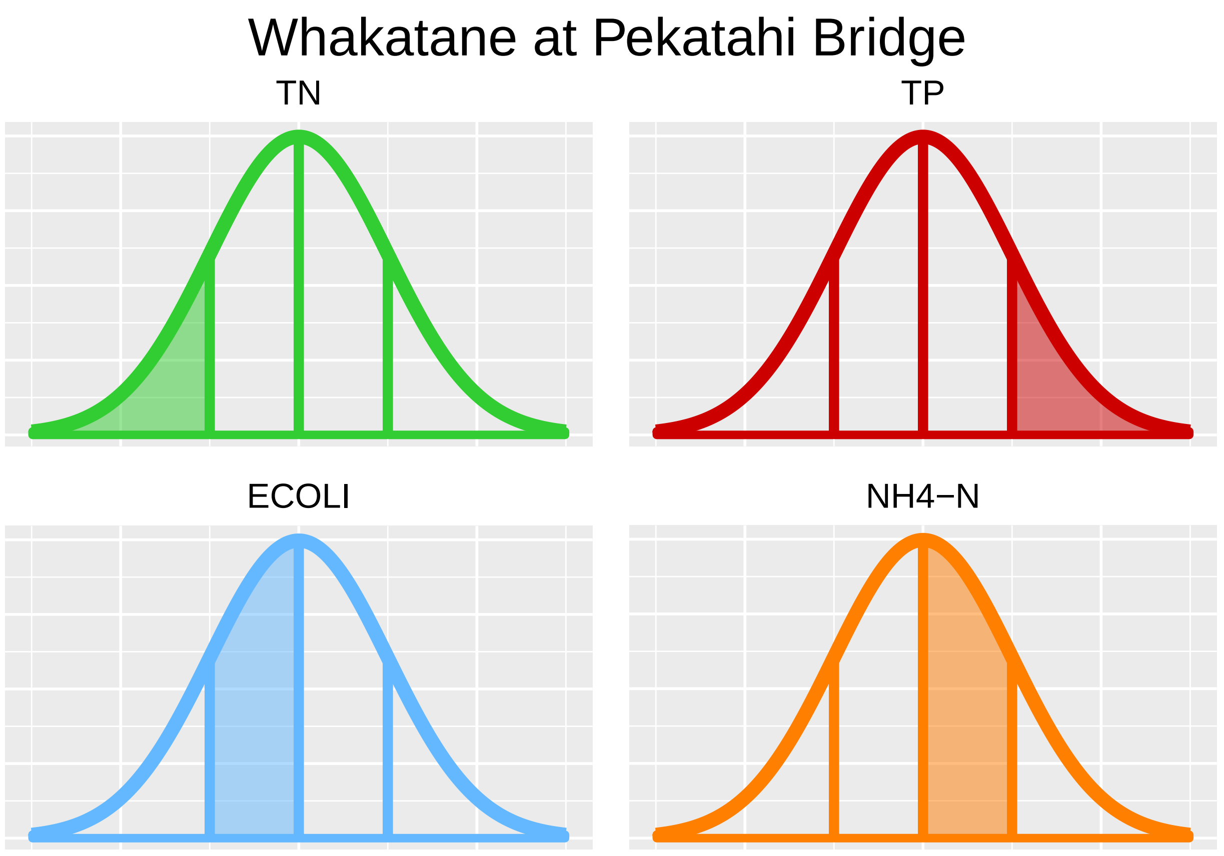
```r
LAWADistPlots(group = 1) + ggtitle("TN") + theme(plot.title = element_text(hjust = 0.5))
```



This is not that useful by itself, but we can add other parameters to give a better 'site description'. Lets just assume that we used the ecdf function to calculate the percentiles of four other parameters. Generally we would do this by summarising the WQ dataset into means for each parameter, and running through each parameter-distribution pair using the ecdf function. We would then create another column where we would categorise the percentile outputs into groups that could be fed into the LAWADistPlots function. In this case, we will assume that: TN = group 2, TP = group 4, ECOLI = group 1, and NH4-N = group 3. We can plot these together using the gridextra package and the code below:

```
# create individual plots and save them as objects
pTN <- LAWADistPlots(group = 1) + ggtitle("TN") + theme(plot.title = element_text(hjust = 0.5))
pTP <- LAWADistPlots(group = 4) + ggtitle("TP") + theme(plot.title = element_text(hjust = 0.5))
pECOLI <- LAWADistPlots(group = 2) + ggtitle("ECOLI") +
    theme(plot.title = element_text(hjust = 0.5))
pNH4 <- LAWADistPlots(group = 3) + ggtitle("NH4-N") +
    theme(plot.title = element_text(hjust = 0.5))

# load the gridExtra and grid package.  These help
# with manipulating the objects in the plotting
# space.
library(gridExtra)
library(grid)
# combine all plots into one
grid.arrange(pTN, pTP, pECOLI, pNH4, ncol = 2, top = textGrob("Whakatane at Pekatahi Bridge",
    gp = gpar(fontsize = 20, font = 1)))
```



The Hazen.Percentile Function

This is an r version of the Hazen calculator on the MFE website: http://www.mfe.govt.nz/publications/fresh-water/bathewatch-user-guide/hazen-percentile-calculator.

Input is in the form of a vector of numbers that you want to calculate the percentile from, and the percentile value that you are interested in.

We can use ECOLI values from the WQData_Appended dataset to run this analysis. Assume we are interested in a 95th percentile value.

```r
Hazen.Percentile(WQData_Appended[, 6], 95)
```

```
## [1] 1274
```

The result is 1274.

The Convert.Columns Function

This function is useful for rapidly converting columns in a data frame to a specific class (e.g. numeric).

Input requirements are:

- **x** - a dataframe
- **y** - a string of desired classes that correspond to the columns in x. Options are currently limited to: “numeric”,“character”,“factor”,or “integer”.

```
# create dummy dataframe
df = data.frame(SiteID = as.character(c(paste(10112:10121,
    rep(c("A", "B"), 5), sep = ""))), y = as.character(c(1:10)),
    z = as.character(c(11:20)), group = as.character(rep(1:5,
        2)), year = as.character(c(1995:2004)), stringsAsFactors = FALSE)

# determine the class of each column
sapply(df, class)
```

```
##      SiteID           y           z       group        year
## "character" "character" "character" "character" "character"
```

All columns are of class 'character'.

```
# convert columns to desired results
df <- Convert.Columns(df, c("factor", "integer", "numeric",
    "factor", "integer"))

# recheck classes
sapply(df, class)
```

```
##    SiteID         y         z     group      year
##  "factor" "integer" "numeric"  "factor" "integer"
```

They are now the classes specified above.

The CheckTrends Function

This function is modified from the code of Sean Hodges (Horizons Regional Council), and is used for calculating trends for the LAWA site. I have amended this slightly so that Aquarius datasets can easily be checked.

We will use the WQData_Appended dataset for this example.

The CheckTrends function requires the following inputs:

- **data** - an AQMultiExtract dataframe, or a data frame with an identical layout.
- **paramcol** - the column of the parameter you are interested in.
- **start** - (optional) start date of analysis in YYYY-MM-DD.
- **end** - (optional) end date of analysis in YYYY-MM-DD.
- **bdisc** - (optional) a boolean to determine if the parameter is black disc (hence a decreasing trend is bad).

Here's an example.

```
CheckTrends(WQData_Appended, paramcol = 7, start = "2010-01-01",
    end = "2017-01-01")
```

```
## $sen.slope
## [1] 0.002958333
##
## $sen.slope.pct
## [1] 1.826131
##
## $p.value
## [1] 0.2958129
##
## $miss
##   1   2   3   4   5   6   7   8   9  10  11  12
## 0.0 0.0 0.0 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##
## $trendscore
## [1] 0
```

As you can see, the output provides a sen slope (sen.slope), percentage annual change (sen.slope.pct), p value (p.value), information about missing data for each month (miss), and a trend score (trendscore). Trend scores at the 95% confidence level are as follows: * **-2** = declining trend by more than 1% per annum. * **-1** = declining trend by less than 1% per annum. * **0** = no significant trend. * **1** = improving trend by less than 1% per annum. * **2** = improving trend by more than 1% per annum.

Trend scores do not change direction for black disc.

Note that this analysis does not adjust for flow. I am still trying to figure out how to do this.