The flowPhyto Package

Francois Ribalet
ribalet@u.washington.edu

David Schruth
dschruth@u.washington.edu

February 5, 2013

1 Licensing

1. Ribalet, F., Schruth, D., Armbrust, E.V. flowPhyto: enabling automated analysis of microscopic algae from continuous flow cytometric data. 2011 *Bioinformatics*, doi: 10.1093/bioinformatics/btr003.

2 Installation

2.1 Unix/Linux/Mac

Building the *flowPhyto* package from source requires that you have a C compiler, and all of the prerequisites for the underlying flowCore package: namely the GNU Scientific library (GSL), and the Basic Linear Algebra Subprograms (BLAS). After these prerequisites are taken care of, the package is ready to install via:

```
R CMD INSTALL flowPhyto_x.y.z.tar.gz
```

After a successful installation the package can be loaded in the normal way: by starting R and invoking the `library` command like so:

```
> library(flowPhyto)
```

2.2 Windows

The *flowPhyto* package is compatible with the Windows version of R and the same prerequisites apply. However, the `pipeline` function and the downstream file-based functions which deploy the four analysis steps to a cluster are not currently supported.

3 Introduction

Flow cytometry is a widely used technique among biologists to study the abundances of populations of microscopic algae living in aquatic environments. A new generation of high-frequency flow cytometer, known as SeaFlow, collects up to several hundred samples per day and can run continuously for several weeks (see Ribalet *et al.*, 2010 for more details). Automated computational methods are needed to analyze the different phytoplankton populations present in each sample. Here we describe the *flowPhyto* R package which performs aggregate statistics on virtually unlimited collections of raw flow cytometry files in a memory efficient, parallelized fashion.

4 The SeaFlow Respository

SeaFlow data are stored in a custom binary file (EVT file) created every 3 minutes and consist of eight 16-bit integer channels namely:

```
> CHANNEL.CLMNS
```

```
[1] "fsc_small" "fsc_perp"  "fsc_big"   "pe"
[5] "chl_small" "chl_big"
```

The SeaFlow repository is composed of julian day labeled directories, each containing chronologically-ordered EVT files. The following code shows how to read one of these files into memory:

```
> evt.file.path <- system.file("extdata","seaflow_cruise","2011_001", "2.evt",
+                                  package="flowPhyto")
> evt <- readSeaflow(evt.file.path)
```

5 Core Functions

5.1 OPP Filtration

Unlike a traditional flow cytometer, SeaFlow directly analyzes a raw stream of seawater using two detectors that determine the position of a particle in the focal region of the instrument optical system (Swalwell *et al.*, 2009). The `filter` function selects optimally positioned particles (OPP) in each EVT file that are used to distinguish the different phytoplankton populations.

```
> opp <- filter(evt, notch=1.1)
```

5.2 Cluster Based Classification

Because the characteristics of each phytoplankton population vary according to environmental conditions and instrument settings, a table of customizable parameters (pop.def.tab) is used to define the pre-gating regions and statistical priors of phytoplankton population clusters.

```
> opp.path <- system.file("extdata","seaflow_cruise","2011_001", "2.evt.opp",
+                                 package="flowPhyto")
> pop.def.path <- system.file("extdata","seaflow_cruise","pop.def.tab",
+                                 package="flowPhyto")
> opp <- readSeaflow(opp.path)
> def <- readPopDef(pop.def.path)
> def
```

```
          abrev            title  xmin  ymin  xmax  ymax
beads     beads            Beads 10000 30000 65000 65000
synecho synecho    Synechococcus  7000  7000 35000 35000
crypto   crypto Cryptophyte-like 30000 30000 65000 65000
diatoms diatoms    Pennates-like 20000 20000 65000 65000
ultra     ultra    Ultraplankton 25000 30000 40000 45000
nano       nano     Nanoplankton 40000 20000 65000 65000
pico       pico     Picoplankton 10000 15000 30000 35000
unknown unknown          Unknown 40000     0 65000 20000
               color      xvar      yvar u.co    lim
beads          black chl_small        pe 0.05  15000
synecho         tan2        pe chl_small 0.25 -10000
crypto       tomato3        pe chl_small 0.75  -1000
diatoms         gold fsc_small   chl_big 0.75  -5000
ultra     palegreen3 fsc_small chl_small 0.50     NA
nano        darkcyan fsc_small   chl_big 0.75     NA
pico    lightseagreen fsc_small chl_small 0.75     NA
unknown          grey fsc_small   chl_big 0.75     NA
```

Above we can see the default population definition table with the two dimentional pregating ranges and the parameters passed to the statistical clustering methods of the *flowMeans* package (Aghaeepour *et al.* 2011).

Below, the `classify` function uses these pre-defined parameters and inputs one or more OPP files (3 by default) to classify individual phytoplankton cells into different populations.

```
> pop <- classify(x=opp, pop.def= def)
```

```
[1] "Clustering 8 populations defined in pop.def table..."
```

```
> table(pop$pop)
```

```
     0       1       2   beads  crypto    nano synecho
   219    3810     178     415      11      39      52
 ultra unknown
   209      67
```

The `plotCytogram` function outputs a series of customizable 2-D cytograms to visualize the phytoplankton populations identified by the `classify` function.

```
> plotCytogram(pop, "fsc_small","chl_small", pop.def= def, add.legend=TRUE, cex=1)
>
```

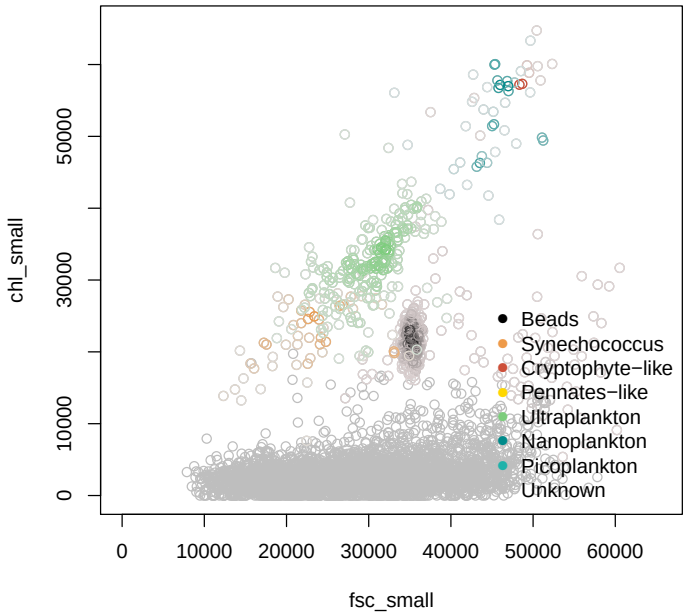

Figure 1: The above 2-D Cytogram depicts the phytoplankton population present in the sample.

5.3 Consensus and Census

`classify` outputs vector files (consensus.vct) that contain the population identification of the cells. `classify` is run in single file increments to provide multiple passes over a single cell and strengthen the clustering analysis. During the `census` step, these multiple-pass vector files are collapsed into one consensus vector, which represents the most likely population classification of the different phytoplankton cells. In addition, `census` produces a one-row census tab file that contains the number of cells per population for each file. The concatenation of these census tab files is used to create a per-population resampling scheme that calculates the number of OPP files necessary so a sufficient number of cells (500 by default) is present in the resampled population.

```
> vct.paths <- sapply(c(1,439,440), function(i)
+                 system.file("extdata","seaflow_cruise","2011_001",
+                         paste("1.evt.opp.",i,'-class.vct',sep=''),
+                                 package="flowPhyto"))
> mat <- do.call(cbind,lapply(vct.paths, read.delim))
> consen.df <- consensus(mtrx=mat)
> table(consen.df$pop)

  beads    nano    pico synecho   ultra       x
     52      25      31      74     174    4644

> aggregate(consen.df$support,list(consen.df$pop), mean)

  Group.1        x
1   beads 2.923077
2    nano 2.960000
3    pico 2.774194
4 synecho 2.959459
5   ultra 2.977011
6       x 2.986865
```

Above is a table of cross tabulated sums per population of the generated consensus vector and a corresponding table of the average 'support' counts. The support column in the output of `consensus` keeps track of the number of the multiple-pass classification vectors that called an event as this population.

Compare the above population count cross tabulation with the output of census below.

```
> census(v=pop$pop, pop.def=def)

  beads synecho  crypto diatoms   ultra    nano    pico
    415      52      11       0     209      39       0
unknown       x
     67       0
```

5.4 Aggregate Statistics

The `summarize` function performs per-population aggregate statistics (cell concentration and the mean and standard deviation of the different channels) using the resampling scheme.

```
> filter.df <- readSeaflow(opp.path, add.yearday.file=TRUE)
> classed <- cbind.data.frame(filter.df, consen.df)
> names(opp.path) <- getFileNumber(opp.path)
> class.jn <- joinSDS(classed, opp.path)

[1] "No fluorescence data found"

> nrow.opp <- sapply(opp.path, function(p) readSeaflow(              p , count.only=TRUE))
> nrow.evt <- sapply(opp.path, function(p) readSeaflow(sub('.opp','',p), count.only=TRUE))
> class.jn$opp <- rep(nrow.opp, times=nrow.opp)
> class.jn$evt <- rep(nrow.evt, times=nrow.opp)
> summarize(class.jn,  opp.paths.str=opp.path)

             day file     pop
x       2011_001    2       x
ultra   2011_001    2   ultra
synecho 2011_001    2 synecho
beads   2011_001    2   beads
pico    2011_001    2    pico
nano    2011_001    2    nano

x       /private/var/folders/pj/7ymf8lld5jx7394cqmb4cwwm0000gn/T/RtmpHz29pl/Rinstaea959ecb1d
ultra   /private/var/folders/pj/7ymf8lld5jx7394cqmb4cwwm0000gn/T/RtmpHz29pl/Rinstaea959ecb1d
synecho /private/var/folders/pj/7ymf8lld5jx7394cqmb4cwwm0000gn/T/RtmpHz29pl/Rinstaea959ecb1d
beads   /private/var/folders/pj/7ymf8lld5jx7394cqmb4cwwm0000gn/T/RtmpHz29pl/Rinstaea959ecb1d
pico    /private/var/folders/pj/7ymf8lld5jx7394cqmb4cwwm0000gn/T/RtmpHz29pl/Rinstaea959ecb1d
nano    /private/var/folders/pj/7ymf8lld5jx7394cqmb4cwwm0000gn/T/RtmpHz29pl/Rinstaea959ecb1d
                     time      lat      long     flow
x       2009-11-09 00:11:24 48.02425 -122.6206 2473.903
ultra   2009-11-09 00:11:24 48.02425 -122.6206 2473.903
synecho 2009-11-09 00:11:24 48.02425 -122.6206 2473.903
beads   2009-11-09 00:11:24 48.02425 -122.6206 2473.903
pico    2009-11-09 00:11:24 48.02425 -122.6206 2473.903
nano    2009-11-09 00:11:24 48.02425 -122.6206 2473.903
        bulk_red salinity temperature event_rate
x         20.538      NaN         NaN       3171
ultra     20.538      NaN         NaN       3171
synecho   20.538      NaN         NaN       3171
beads     20.538      NaN         NaN       3171
pico      20.538      NaN         NaN       3171
nano      20.538      NaN         NaN       3171
```

```
        fluorescence  evt  opp    n   conc fsc_small_mean
x                NaN 5000 5000 4644 0.6257       30064.09
ultra            NaN 5000 5000  174 0.0234       29088.79
synecho          NaN 5000 5000   74 0.0100       29839.01
beads            NaN 5000 5000   52 0.0070       30441.90
pico             NaN 5000 5000   31 0.0042       32203.87
nano             NaN 5000 5000   25 0.0034       29994.32
        fsc_small_median fsc_small_sd fsc_small_mode
x                30671.0     9007.538       34771.53
ultra            28356.0     8937.524       32092.49
synecho          30908.0     9141.493       34616.53
beads            30598.5     9867.279       33449.51
pico             32400.0     8736.306       33824.52
nano             29368.0    11244.956       36267.55
        fsc_small_width fsc_small_npeaks fsc_perp_mean
x              16207.25                1      29077.59
ultra          22720.35                1      28021.56
synecho        25179.38                1      28962.26
beads          22475.34                1      29857.33
pico           17145.26                1      31374.48
nano           34726.53                1      28703.64
        fsc_perp_median fsc_perp_sd fsc_perp_mode
x                29684.0    9217.751      33802.52
ultra            27774.5    9093.033      28022.43
synecho          29295.0    9086.784      32978.50
beads            29571.0    9492.266      31636.48
pico             31643.0    8945.106      33819.52
nano             28133.0   10762.561      32325.49
        fsc_perp_width fsc_perp_npeaks fsc_big_mean
x             15884.24               1            0
ultra         21818.33               1            0
synecho       25680.39               1            0
beads         20600.31               1            0
pico          20240.31               1            0
nano          31697.48               1            0
        fsc_big_median fsc_big_sd fsc_big_mode
x                    0          0            0
ultra                0          0            0
synecho              0          0            0
beads                0          0            0
pico                 0          0            0
nano                 0          0            0
        fsc_big_width fsc_big_npeaks  pe_mean pe_median
x                   0              1 4986.835       992
ultra               0              1 3277.862       924
synecho             1              1 2763.365      1032
```

```
beads               0              1 4688.750       917
pico                0              1 2191.419      1043
nano                0              1 4505.800       704
            pe_sd   pe_mode pe_width pe_npeaks
x       12190.171 43565.665  965.015         2
ultra    9397.069   707.011 1266.019         1
synecho  8462.308   963.015 1489.023         1
beads   12000.380   599.009 1243.019         1
pico     4741.385  1044.016 1034.016         1
nano    12565.295   589.009 1261.019         1
        chl_small_mean chl_small_median chl_small_sd
x             6570.844           2865.5     9495.711
ultra         6069.322           2902.5     9673.495
synecho       4327.676           2882.5     5564.897
beads         8068.654           3213.5    11935.362
pico          7863.581           2704.0    12587.039
nano          7090.720           2771.0    10168.323
        chl_small_mode chl_small_width chl_small_npeaks
x             2437.037        3331.051                1
ultra         2345.036        4106.063                1
synecho       2045.031        4087.062                1
beads         2664.041        3557.054                1
pico          2262.035        3297.050                1
nano          1795.027        4386.067                1
        chl_big_mean chl_big_median chl_big_sd chl_big_mode
x           4254.312              0   6413.913       41.001
ultra       4224.092              0   6986.534      108.002
synecho     2572.541              0   4582.520       20.000
beads       5332.308            568   8221.586      226.003
pico        5024.516              0   8796.232      347.005
nano        4947.200              0   6885.855      533.008
        chl_big_width chl_big_npeaks
x            1215.019              1
ultra        2248.034              1
synecho      1554.024              1
beads        4514.069              1
pico         4526.069              1
nano         5862.089              1
```

The summarize function associates the corresponding acquisition time and location (latitude and longitude). It outputs a summary table of the entire set of SeaFlow data.

The `plotStatMap` creates customizable plots of the geo-referenced data created by `summarize`. A combination of the different parameters per population or a single parameter over different populations can be selected depending on the purpose of the analysis.

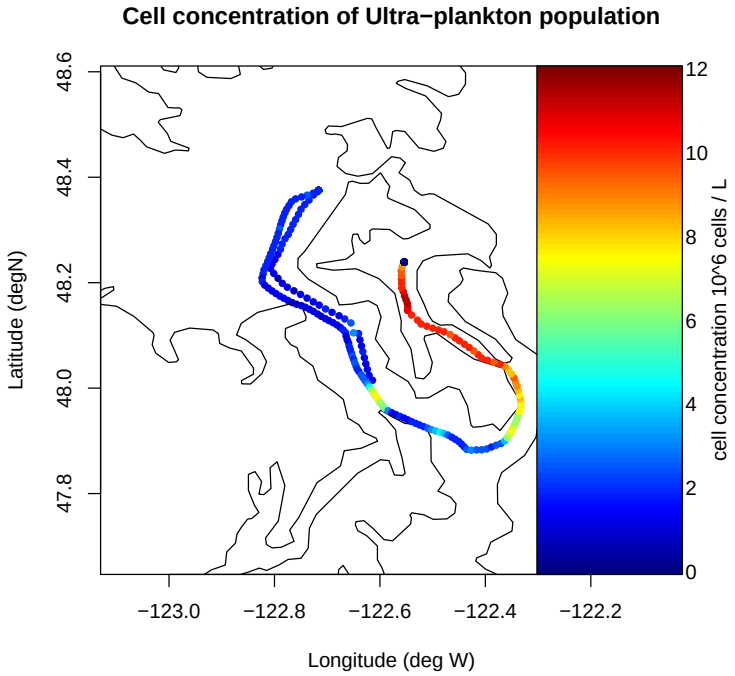

Figure 2: Ultra-plankton concentration for the Puget Sound in November, 2009

```
> stat.tab <- system.file("extdata","seaflow_cruise","stats.tab",
+                                          package="flowPhyto")
> stats <- read.delim(stat.tab)
> plotStatMap(df=stats, pop='ultra', z.param='conc', margin=0.2, zlab=expression(paste('Cell
+ main="Cell concentration of Ultra-plankton population")
> mtext(line=1, side=4, "cell concentration 10^6 cells / L")
>
```

6 File-Based Functions, Input Validation, and The Pipeline

Each of the above core functions has a file based analog that takes one (or several) paths as it's main input parameter and outputs one or many files. For examples of these please check the man pages for individual file functions.

6.1 Directory Initialization

First you'll need transfer the data to a location where there is plenty of extra disk space (around 25 percent more space than the raw EVT files alone). Then you'll want to make sure the directory has write access for the user who will be running the pipeline. Changing your working directory to the output directory is also recommended as many of the cruise specific job files get written there by default. Additional steps such as creating a log or plot sub directory in the repository or a new record in your cruise database (if you plan on uploading the resulting statistics or sds information to your database) may be desirable as well.

6.2 SDS and pop.def.tab validation

One of the more important pre-processing steps to make can be with validating the SDS file before running the pipeline. One should both check for evidence of parsing errors that may have crept into the ship's data stream. The following code demonstrates one way this could be done, namely, longitude and latitude checking:

```
> path <- system.file("extdata","seaflow_cruise",package="flowPhyto")
> sds <- combineSdsFiles(path)
> plot(sds$LON, sds$LAT)
```

Additionally any externally defined population definition table should be validated using the following function.

```
> validatePopDef(readPopDef(pop.def.path))
```

```
[1] TRUE
```

An external pop.def can be specified by placing a file named 'pop.def.tab' in the cruise's directory. The parameter names and data types should match those found in the POP.DEF object. If such a file is not present, one will get created automatically from the dataframe hard coded into Define.R.

6.3 Running the Pipeline

The pipeline itself is merely a cluster deployment function which executes, in concerted batches, each of the file-based wrapper functions for the 4 main

analysis steps. Many of the sub-function specific parameters can also be passed through from this upper level function. The following example copies the very small bundled example data set to the present working directory and runs the pipeline for just step 4 which calculate statistics on a repository that has already undergone analysis steps 1 through 3.

```
> example.cruise.name <- 'seaflow_cruise'
> temp.out.dir <- '.' #path.expand('~')
> output.path <- paste(temp.out.dir,'/',example.cruise.name,sep='')
> seaflow.path <- system.file("extdata", example.cruise.name, package="flowPhyto")
> file.copy(from=seaflow.path, to=temp.out.dir, recursive=TRUE)

[1] TRUE

> pipeline(repo= temp.out.dir, cruise.name='seaflow_cruise', steps=4, parallel=FALSE)
> unlink(example.cruise.name, recursive=TRUE)
```

The most important parameters to set when calling pipeline are 'repo' which should be set to the location of your repository, and 'parallel' which tells the function whether or not to run in serial or parallel. Currently parallel jobs are simply submitted a via a cluster submission command such as 'qsub' (for Torque/SGE) or 'mosrun' (for MOSIX) as specified by the 'submit.cmd' option. For the purposes of this brief example 'parallel' has been set to FALSE but should almost always, where possible, be set to TRUE (the default) when running the pipeline over realistically sized, day or more long data sets. Additionally, the submit.cmd parameter was set to use qsub as a non-functional example. (Normally parallel=FALSE and submit.cmd would not be used together). Future plans for parallelization include replacement of the above 'R CMD BATCH' and 'submit.cmd' based parallelization with a PVM/MPI based *snow* package implementation.

6.4 Cleanup

There are two useful functions that can help to clean up the aftermath of all of the pipelined R CMD BATCH calls. The `cleanupLogs` function deletes log files depending on their error status. The `clearOutput` removes any output files from specified steps to clear the way for a re-run of the pipeline.

7 Example Dataset

The examples bundled with this dataset have been artifically reduced both in size and in number to make the package as light weight as possible. For a more realistic example you can visit our website `http://seaflow.ocean.washington.edu` to download a copy of the day-long 2009 Puget Sound cruise.

References

Aghaeepour, N. *et al.* (2011) Rapid cell population identification in flow cytometry data. *Cytometry Part A*, **10**A(79):6–13.

Ribalet,F. *et al.* (2010) Unveiling a phytoplankton hotspot at a narrow boundary between coastal and offshore waters. *Proc. Nat. Acad. Sci.*, **107**(38):16571–16576.

Ribalet, F., Schruth, D., Armbrust, E.V. flowPhyto: enabling automated analysis of microscopic algae from continuous flow cytometric data. *Bioinformatics*, submitted.

Spidlen,J. *et al.* (2010) Data file standard for flow cytometry, version fcs 3.1. *Cytometry Part A*, **77**A(1):97–100.

Swalwell,J. *et al.* (2009) Virtual-core flow cytometry. *Cytometry Part A*, **75**A(11):960–965.