flowPhyto

October 25, 2011

`CHANNEL.CLMNS` *Seaflow detector column names*

Description

Vector of field names for the flow cytometry detector channels of an event [EVT] or [OPP] dataframe.

Examples

```
CHANNEL.CLMNS
```

`CHANNEL.CLMNS.SM` *Seaflow detector column names*

Description

Vector of field names for the flow cytometry detector channels of an event [EVT] or [OPP] dataframe (just the SMall columns).

Examples

```
CHANNEL.CLMNS.SM
```

`EVT.HEADER` *Seaflow File Headers*

Description

Vector of field names for the columns of an event [EVT] or [OPP] dataframe.

Examples

```
EVT.HEADER
```

POP.DEF *population definition table*

Description

The pop.def table is used to pre-gate and guide the clustering analysis in the classify step. This one is a hard coded dataframe that is used by default in the analysis if one is not specified by the user.

Examples

```
POP.DEF
```

REPO.PATH *Seaflow Data directory*

Description

Global string indicating the location of the directory that contains all of the data (SeaFlow) repositories

Examples

```
REPO.PATH
```

census *Cross tabulate a consensus vector*

Description

Cross tabulate the population composition

Usage

```
census(v, pop.def)
```

Arguments

v	a consensus vector of population classifications.
pop.def	A population (rows) definition dataframe with parameters (columns) for gating and clustersing.

Value

a one row, cross-tabulated dataframe of counts with one column for each population specified by the rows in the pop.def dataframe. Zeros are filled in for absent populations.

Examples

```
opp.file.path <- system.file("extdata","seaflow_cruise","2011_001", "1.evt.opp",
package="flowPhyto")
pop.file.path <- system.file("extdata","seaflow_cruise","pop.def.tab",
package="flowPhyto")

opp <- readSeaflow(opp.file.path)
def <- readPopDef(pop.file.path)
pop <- classify(x=opp, pop.def= def)

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

---

censusFile *Create a consensus from several classification vector files and cross*

Description

Create a consensus from several classification vectors and cross tabulate the population composition

Usage

```
censusFile(opp.path, map.margin=2, output.path=getCruisePath(opp.path),
def.path=paste(getCruisePath(opp.path),'pop.def.tab',sep=''))
```

Arguments

`opp.path`	Path to OPP event file.
`map.margin`	Margin in latitude/longitude around the map plots.
`output.path`	Path to the directory where you wish to output data.
`def.path`	Path to the file that defines how to gate & cluster the events into populations.

Value

a one dimentional consensus vector file (one column) and a one dimentional cross tabulation file (one row) both writen to disk

Examples

```
example.cruise.name <- 'seaflow_cruise'
temp.dir <- '.'

seaflow.path <- system.file("extdata", example.cruise.name, package="flowPhyto")
file.copy(from=seaflow.path, to=temp.dir, recursive=TRUE)

#opp.in.path <- system.file("extdata","seaflow_cruise","2011_001", "1.evt.opp",
# package="flowPhyto")

opp.out.path <- paste(temp.dir,'/',example.cruise.name,'/2011_001/1.evt.opp',sep='')
censusFile(opp.path=opp.out.path, map.margin=.5)
unlink(example.cruise.name, recursive=TRUE)
```

`classify` *Cluster the different Phytoplankton Populations*

Description

Classify the different cell populations from an OPP or FCS dataframe according to a pre-defined parameters of population definition

Because the characteristics of each phytoplankton populations varied according to environmental conditions and instrument settings, a customizable table of pre-defined parameters (pop.def) is used to help in gating the different phytoplankton populations. The rows of the pop.def table represent the names of the different populations. The columns of the pop.def table represent the parameters used for gating and clustering the different populations. The function uses these pre-defined parameters and inputs a single OPP or FCS file to cluster cell populations using either flowClust or flowMeans package

Usage

```
classify(x, pop.def=POP.DEF, func=2, varnames = CHANNEL.CLMNS.SM, numc=0, noise=
```

Arguments

`x`	an OPP or FCS dataframe.
`pop.def`	pop.def table that defines how to gate & cluster the events into populations.
`func`	Choose the clustering method, either flowClust (func = 1) or flowMeans (func = 2, by default) function
`varnames`	A character vector specifying the variables (columns) to be included in clustering when choosing flowMeans.
`numc`	Number of clusters when choosing flowMeans. If set to 0 (default) the value matches the number of populations defined in pop.def table . If set to NA, the optimal number of clusters will be estimated automatically.
`noise`	Set up the noise threshold for phytoplankton cells. Only cells with chlorophyll value higher than the noise will be clustered
`plot.cluster`	Plot the output of clustering when choosing flowMeans
`plot.assignment`	Plot the output of Matching cluster number with cell population defined in pop.def.tab when choosing flowMeans
`try.context`	Default value set up to 'local'
`...`	additional arguments to be passed to the plot function

Value

an OPP or FCS dataframe like the input x but with an additional column 'pop' indicating population assignment

Examples

```
## reading a standard SeaFlow file
opp.path <- system.file("extdata","seaflow_cruise","2011_001", "1.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)
pop <- classify(x=opp, pop.def= def)

table(pop$pop)

## reading from a fcs file format
fcs.file.path <- system.file("extdata","fcs_cruise", "CD.20070615.A.0010.fcs",
package="flowPhyto")
pop.def.path2 <- system.file("extdata","fcs_cruise", "pop.def.tab",
package="flowPhyto")

require(flowClust)
ff <- read.FCS(fcs.file.path, transformation=FALSE)
df <- caroline::tab2df(exprs(ff))
names(df) <- EVT.HEADER[c(2,6,5,8,7,NA,9,NA,NA,10,1)]

#if(!all(EVT.HEADER[5:length(EVT.HEADER)] %in% names(df)))
#  warning('Match your column names to the EVENT.COLUMN.NAMES')
def2 <- readPopDef(pop.def.path2)
pop2 <- classify(x=df, pop.def= def2, func=1)

table(pop2$pop)

#plotCytogram(pop2, 'fsc_small', 'chl_small', pop.def=def2)

#optionally write this exported dataframe to disk as an opp file
#writeSeaflow('converted.fcs.evt', df)
```

`classifyFile` *Cluster the different Phytoplankton Populations*

Description

classify the different cell populations from an OPP or FCS file according to a pre-defined parameters of population definition and output a consensus vector.

For each group of OPP or FCS file, the function outputs a single vector file (consensus.vct) that contains the population identification of each single cell. The function is run in single OPP or FCS file increments to provide multiple vector files over each OPP or FCS file and strengthen the clustering analysis.

Usage

```
classifyFile(opp.path, concat.ct=3, output.path=getCruisePath(opp.path),
 def.path=paste(getCruisePath(opp.path),'pop.def.tab',sep=''), func=2, varnames
```

Arguments

opp.path	Path to OPP file.
concat.ct	Number of OPP files to concatenate at a time.
output.path	Path to the directory where you wish to output data.
func	Choose between Classify (func = 1) or Classify2 (func = 2, by default) function
varnames	A character vector specifying the variables (columns) to be included in clustering when choosing flowMeans.
numc	Number of clusters when choosing flowMeans. If set to 0 (default) the value matches the number of populations defined in pop.def table . If set to NA, the optimal number of clusters will be estimated automatically.
noise	Set up the noise threshold for phytoplankton cells. Only cells with chlorophyll value higher than the noise will be clustered
def.path	Path to the file that defines how to gate & cluster the events into populations.

Value

a consensus vector file composed of one single column indicating population assignment of each event from the OPP or FCS file.

Examples

```
example.cruise.name <- 'seaflow_cruise'
temp.out.dir <- '.'

seaflow.path <- system.file("extdata", example.cruise.name, package="flowPhyto")
file.copy(from=seaflow.path, to=temp.out.dir, recursive=TRUE)

classifyFile(paste(temp.out.dir,'/',example.cruise.name,'/2011_001/1.evt.opp',sep=''), fu
unlink(example.cruise.name, recursive=TRUE)
```

---

cleanupLogs *Remove R Batch Output Files.*

Description

Each step of the flowPhyto pipeline generates a large number of .Rout files that should be cleaned up. This function does just that and optionally leaves the outputs with errors intact for troubleshooting purposes.

Usage

```
cleanupLogs(log.dir='.', keep.erred=TRUE)
```

Arguments

log.dir	The directory where the log files were written to.
keep.erred	Whether or not to keep the log files with errors.

Value

none

Examples

```
cleanupLogs('.')
```

`clearOutputs` *Clear the output files from particular flowPhyto pipeline steps.*

Description

Each step of the flowPhyto pipeline generates many files which are used as indicators of completion. This function helps to clear those files away to allow for proper waiting between steps of a pipeline rerun.

Usage

```
clearOutputs(cruise.path='.', steps=1:4)
```

Arguments

`cruise.path`	Path to the cruise directory.
`steps`	Steps for which to clear the outputs.

Value

none

Examples

```
example.cruise.name <- 'seaflow_cruise'
temp.out.dir <- '.'

seaflow.path <- system.file("extdata", example.cruise.name, package="flowPhyto")
file.copy(from=seaflow.path, to=temp.out.dir, recursive=TRUE)

clearOutputs(paste(temp.out.dir,'/',example.cruise.name,sep=''), steps=3)
unlink(example.cruise.name, recursive=TRUE)
```

combineCensusFiles *Combine the Census Files.*

Description

The census step generates a single row of population counts per EVT file. This function collects and concatenates these into a single dataframe.

Usage

```
combineCensusFiles(cruise.dir='.')
```

Arguments

cruise.dir	Path the cruise directory.

Value

a dataframe of counts per EVT file (rows) and per population (columns)

Examples

```
seaflow.path <- system.file("extdata", 'seaflow_cruise', package="flowPhyto")
census <- combineCensusFiles(seaflow.path)
census
```

combineSdsFiles *Combine the SDS files.*

Description

The SDS files in each directory are concatenated into one dataframe via this function.

Usage

```
combineSdsFiles(cruise.dir='.')
```

Arguments

cruise.dir	Path to the cruise directory.

Value

A data frame representing the concatenation of all cruise subdirectory SDS files.

Examples

```
seaflow.path <- system.file("extdata", 'seaflow_cruise', package="flowPhyto")
sds <- combineSdsFiles(seaflow.path)
sds
```

consensus *create a consensus vector from several classification vectors*

Description

Create a consensus from several classification vectors

Usage

```
consensus(mtrx, threshold=.6)
```

Arguments

mtrx	a matrix of n concatenated classification vectors.
threshold	the minimum percentage of identical classification calls required for an unambiguous consensus call.

Value

a vector the same length as the number of rows in the input matrix with a population classification call for each element

Examples

```
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))
v <- consensus(mtrx=mat)
table(v$pop)
aggregate(v$support,list(v$pop), mean)
```

consensusFile *Create a consensus from several classification vector files and cross*

Description

Create a consensus from several classification vectors and cross tabulate the population composition

Usage

```
consensusFile(opp.path, pattern='.[0-9]+-class.vct$',
output.path= paste(.createOutputPath(opp.path,
getCruisePath(opp.path)), ".consensus.vct", sep =""))
```

Arguments

opp.path	Path to OPP event file.
pattern	The suffix regular expression pattern used to find the n pass vector files for this opp file
output.path	Path to the directory where you wish to output data.

Value

a one dimentional consensus vector file on disk and an invisible vector in memory

Examples

```
cruise.nm <- 'seaflow_cruise'
temp.out.dir <- '.'

seaflow.path <- system.file("extdata", cruise.nm, package="flowPhyto")
file.copy(from=seaflow.path, to=temp.out.dir, recursive=TRUE)

opp.file.path <- system.file("extdata","seaflow_cruise","2011_001", "1.evt.opp",
package="flowPhyto")

consensusFile(opp.path=paste(temp.out.dir,'/',cruise.nm,'/2011_001/1.evt.opp',sep=''))
unlink(cruise.nm, recursive=TRUE)
```

---

createResamplingScheme

Generate the Population Resampling scheme.

Description

.

Usage

```
createResamplingScheme(cruise.path, resample.min=500, resamp.concat.max = 5)
```

Arguments

cruise.path	Path cruise directory.
resample.min	Minimum number of cells required for a population's resampling.
resamp.concat.max	Maximum number of files allowed to be concatenated to get the resample minimum.

Value

a vector of comma delimted population named by the corresponding comma delimted year_day/file.

Examples

```
seaflow.path <- system.file("extdata", 'seaflow_cruise', package="flowPhyto")
createResamplingScheme(seaflow.path)
```

`filter` *Filter Optimally Positioned Particles from an EVT dataframe*

Description

The function normalizes the signals of the two position-sensitive detectors (D1 and D2) by the forward angle light scatter (fsc_small) signal to identify optimally positioned particles (OPP) from an EVT dataframe. Optionally the function outputs a control quality plot for OPP filtration.

Usage

```
filter(events,  width=1, notch=1, slope=NULL, edge=1, do.plot=FALSE)
```

Arguments

`events`	event dataframe
`slope`	correction factor for the stream alignment. When stream is not properly aligned, aligned particles do not scatter light equally on D1 and D2 and therefore do not lie onto the 1:1 line on the scatter plot of D2 vs D1. By default, the value of the slope is calculated as the ratio of D2/D1
`width`	the width of the gate to the sides of the 1:1 equal detector response defines the allowed error in particle trajectories across the width of the stream.
`notch`	the correction factor for the sensitivity of FSC with respect to D1 and D2. Scattered light from focused particles is maximal at the forward scatter detector (FSC) and minimal at both position detectors. When the sensitivity of FSC and D1/D2 detectors is adjusted to respond equally to focused calibration particles, the FSC normalized by the signal of both position detectors must be lower than 1.
`edge`	location of the boundary layer between water/air. Particles located at the boundary layer scatter light that can be detected by the position detectors.
`do.plot`	create a plot that showed the different steps for filtering out non-optimally positioned particles

Value

a optimal-position filtered event dataframe

Examples

```
evt.file.path <- system.file("extdata","seaflow_cruise","2011_001", "1.evt",
package="flowPhyto")
evt <- readSeaflow(evt.file.path)
opp <- filter(evt)
summary(opp)
```

`filterFile` *Filter an EVT file and output an Optimally Positioned Particles (OPP)*

Description

The function normalizes the signals of the two position-sensitive detectors (D1 and D2) by the forward angle light scatter (fsc_small) signal to identify optimally positioned particles (OPP) from an EVT file

Usage

```
filterFile(evt.path, width=1, notch=1, slope=NULL, edge=1, map.margin=2, output.
```

Arguments

`evt.path`	path to the raw EVT file to be filtered.
`slope`	correction factor for the stream alignment. When stream is not properly aligned, aligned particles do not scatter light equally on D1 and D2 and therefore do not lie onto the 1:1 line on the scatter plot of D2 vs D1. By default, the value of the slope is calculated as the ratio of D2/D1
`width`	the width of the gate to the sides of the 1:1 equal detector response defines the allowed error in particle trajectories across the width of the stream.
`notch`	the correction factor for the sensitivity of FSC with respect to D1 and D2. Scattered light from focused particles is maximal at the forward scatter detector (FSC) and minimal at both position detectors. When the sensitivity of FSC and D1/D2 detectors is adjusted to respond equally to focused calibration particles, the FSC normalized by the signal of both position detectors must be lower than 1.
`edge`	location of the boundary layer between water/air. Particles located at the boundary layer scatter light that can be detected by the position detectors.
`output.path`	path to the directory where you wish to output OPP file.
`map.margin`	margin in latitude/longitude around the map plots.

Value

a seaflow opp evt file and a plot of the filtration process

Examples

```
example.cruise.name <- 'seaflow_cruise'
temp.dir <- '.'
seaflow.path <- system.file("extdata", example.cruise.name, package="flowPhyto")
file.copy(from=seaflow.path, to=temp.dir, recursive=TRUE)

filterFile(paste(temp.dir,'/',example.cruise.name,'/2011_001/1.evt',sep=''), map.margin=.
unlink(example.cruise.name, recursive=TRUE)
```

getCruiseFiles *Get all of the files from a cruise.*

Description

This is a convenience function to grab all of the raw data files from the all of the julian day sub directories in a seaflow cruise directory.

Usage

```
getCruiseFiles(cruise.dir='.', prefix='[0-9]{1,3}', ext='\\.evt', range=NULL)
```

Arguments

cruise.dir	Path to cruise.
prefix	a prefix to add to the files you wish to list.
ext	extension of the files of interest.
range	A named, two-integer vector specifying the start and end (inclusive) range for subsetting the input files used in each analysis step (with the exception of summarize). Values should be a (evt/opp) file numbers and names should be strings corresponding to the year_julianday directory names. The nv() function is useful for creating this vector.

Value

a vector of cruise file names

Examples

```
path <- system.file("extdata","seaflow_cruise", package="flowPhyto")
getCruiseFiles(path)
```

getCruisePath *Get the Cruise Directory.*

Description

Retrieve the cruise directory path from a path string pointing to a file from that cruise

Usage

```
getCruisePath(this.path, slash=TRUE)
```

Arguments

this.path	Path to a file.
slash	Boolean to indicate which if a slash should be used

Value

the cruise path's direcgtory

Examples

```
path <- system.file("extdata","seaflow_cruise","2011_001","1.evt.opp",
package="flowPhyto")
getCruisePath(path)
```

`getFileNumber` *Get the (original) integer file number of any seaflow repository file.*

Description

Each seaflow EVT file is assigned a unique (per-directory) integer which get's carried on to subsequent processing steps. This function extracts that number from any of the original or downstream files.

Usage

```
getFileNumber(file.path)
```

Arguments

`file.path`	Path to the file whose name you wish to extract a number from.

Value

an integer corresponding to the original event file number

Examples

```
path <- system.file("extdata","seaflow_cruise","2011_001","1.evt.opp",
package="flowPhyto")
getFileNumber(path)
```

joinSDS *join this dataframe to the corresponding SDS log file entry.*

Description

perform aggregate statistics on a particular combination of filtered opp or fcs files for a particular population.

Usage

```
joinSDS(x, opp.paths)
```

Arguments

x	a OPP dataframe.
opp.paths	the named paths of the SDS files (names should be character strings of the corresponding file)

Value

a dataframe of the merge between the events and the SDS log info.

Examples

```
opp.path <- system.file("extdata", 'seaflow_cruise','2011_001','1.evt.opp', package="flow
opp <- readSeaflow(opp.path, add.yearday.file=TRUE)

opp.join <- joinSDS(opp, caroline::nv(opp.path, 1))
summary(opp.join)
```

pipeline *Run the SeaFlow Pipeline*

Description

run the pipeline

Usage

```
pipeline(cruise.name='', repo=REPO.PATH, range=NULL, steps=1:4, pct=.97,
clust.concat.ct=3, resample.size=300, resamp.concat.max=10,
 filter.width=1.5, filter.notch=1, filter.edge=1,
 classify.func =2, classify.varnames=CHANNEL.CLMNS.SM, classify.numc=0, classify
 map.margin=2,
concat.sds=!is.na(match(1,steps)), load.to.db=FALSE,  preplot=FALSE, cleanup=TRU
input.path=paste(repo, '/', cruise.name, sep=''),
output.path=input.path,  log.dir=output.path,
def.path=paste(input.path,'/', 'pop.def.tab',sep=''), parallel=TRUE, submit.cmd=
```

Arguments

cruise.name	Simplified cruise name (same name as the subdirectory in the seaflow data dir).
steps	Which steps of the pipeline to run. step 1 is filter, step 2 is classify, step 3 is census and consensus, step 4 is summarize. 1:2 will do step 1 to 2, etc.
pct	percentage completion (number of indicator files created vs input files) each job step should go to.
clust.concat.ct	Number of event file to concatenate at a time during the clustering/classification step.
map.margin	Margin in latitude/longitude around the map plots.
resample.size	Minimum number of events in a population.
resamp.concat.max	Maximum number of allowable event files to concatenate to generate statistics from.
filter.notch	the location of the x=y (by default) point to create the notch in the gated filter
filter.width	the margin of error for particle alignment determination in the filter step.
filter.edge	location of the boundary layer between water/air. Particles located at the boundary layer scatter light that can be detected by the position detectors for the filter step.
classify.func	Choose the clustering method, either flowClust (func = 1) or flowMeans (func = 2, by default) function
classify.varnames	A character vector specifying the variables (columns) to be included in clustering when choosing flowMeans.
classify.numc	Number of clusters when choosing flowMeans. If set to 0 (default) the value matches the number of populations defined in pop.def table . If set to NA, the optimal number of clusters will be estimated automatically.
classify.noise	Set up the noise threshold for phytoplankton cells. Only cells with chlorophyll value higher than the noise will be clustered
concat.sds	Determines if the sds files in the individual julian day directories should be concatenated together into sds.tab
load.to.db	Load the sds and stat files to the database.
preplot	Preplot the level 2 analysis plots to 'output.path'.
cleanup	Cleanup the submission and (non error reporting) R CMD BATCH log files.
input.path	Path to the directory with input data (raw evt or opp files.
output.path	Path to the directory where you wish to output data.
log.dir	Path to the directory where log file will be written.
def.path	Path to the file that defines how to gate & cluster the events into populations.
parallel	Boolean indicating if the job should be run in parallel using qsub (vs in serial)
repo	Full path to your SeaFlow repository

range	A named, two-integer vector specifying the start and end (inclusive) range for subsetting the input files used in each analysis step (with the exception of summarize). Values should be a (evt/opp) file numbers and names should be strings corresponding to the year_julianday directory names. The nv() function is useful for creating this vector.
submit.cmd	the command used to deploy an R CMD BATCH system call to a cluster. Must be used in conjunction with parallel=TRUE.

Examples

```
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)

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

---

plotCruiseStats *Plot a seaflow cruise statistics*

Description

Read in the pop.stats.tab file and plot maps, or line plots of it and optionall sds info

Usage

```
plotCruiseStats(cruise, x.var='map', y.vars=c('conc','chl_small','fsc_small'),
pops= c("ultra", "synecho"), sds.var=NULL,
date.range=as.POSIXct(c("2009-01-01", "2099-12-31"), tz='UTC'),
output.path=paste(REPO.PATH, cruise,'/plots/',sep=''), ...)
```

Arguments

cruise	Simplified cruise name (same name as the subdirectory in the seaflow data dir).
x.var	X variable: Either map or lat, long or time.
y.vars	Y variables: either conc, fluor, or fsc.
pops	Which populattions to plot. See the pop datastructure for abreviations to use.
sds.var	Which of the sds variables to plot as a secondary axis in a line plot.
date.range	date range.
output.path	Path to the directory where you wish to output data.
...	Additional parameters passed to plot.

Value

an overview statistics plot file is output to disk

Examples

```
cruise <- system.file("extdata","seaflow_cruise", package="flowPhyto")

plotCruiseStats(cruise=cruise, output.path='.')
```

---

plotCytogram *Plot a Phytoplankton Cytogram*

Description

Plot a Phytoplankton Cytogram

Usage

```
plotCytogram(df, x.ax, y.ax, add.legend=FALSE, pop.def=POP.DEF, cex=0.5, pch=1,
```

Arguments

df	a dataframe of events (rows) and channels (columns).
x.ax	column to plot in the x.axis
y.ax	column to plot in the y.axis
add.legend	should the plot automatically generate a legend
pop.def	A population (rows) definition dataframe with parameters (columns) for gating and clustersing.
cex	character expansion for the points. Undefined background points are 1/3rd of the foreground points.
pch	point character
xlab	label for the x axis. by default equals x.ax
ylab	label for the y axis. by default equals y.ax
...	other paramters passed to plot

Value

a cytogram plot

Examples

```
opp.file.path <- system.file("extdata","seaflow_cruise","2011_001", "1.evt.opp",
package="flowPhyto")
pop.file.path <- system.file("extdata","seaflow_cruise","pop.def.tab",
package="flowPhyto")

opp <- readSeaflow(opp.file.path)
def <- readPopDef(pop.file.path)
pop <- classify(x=opp, pop.def= def)

# Visualize the result of Classify using the function plotCytogram()
plotCytogram(pop, "fsc_small","chl_small", pop.def= def)
```

plotLatLongMap *plot file location on a map*

Description

plot the location of the file (longitude and latitude) on a map

Usage

```
plotLatLongMap(lat, long, track=NULL, margin=0.1, col='red',
legend=NULL, pch=20, cex=1.5, lwd=1, zlab=NA, xlim=NULL, ylim=NULL, ...)
```

Arguments

lat	vector of latitudes.
long	vecor of longitudes.
track	a dataframe of the cruise track (from the sds file) with longitude (long) and latitude (lat) named accordingly.
margin	margin of longitude and latitude around edges of current position.
col	vector of colors of the cruise track.
legend	vector of named colors for use in a heatmap color legend.
pch	point character
cex	character expansion
lwd	line width
zlab	the label for the values of the heatmap color gradient passed as 'col'
xlim	limit of x axis
ylim	limit of y axis
...	additional arguments to be passed to the plot function

Examples

```
stat.tab <- system.file("extdata","seaflow_cruise","stats.tab",
package="flowPhyto")
stats <- read.delim(stat.tab)
plotLatLongMap(stats$lat[10], stats$long[10], track=stats)
```

plotStatMap *plot the stat.tab file on a map*

Description

plot the result of on a map

Usage

```
plotStatMap(df, pop, z.param, margin = 0.1, zlab = z.param, ma = 1, xlim= NULL,
```

Arguments

df	dataframe of the summary data, ie stat.tab
pop	Name should match the one written in the stats.tab created by the Summarize function
z.param	parameter in the dataframe for which to plot the heatmap
zlab	the label for the values of the heatmap color gradient passed as 'col'
margin	margin of longitude and latitude around edges of current position.
ma	number of periods to average over z.param
xlim	limit of x axis
ylim	limit of y axis
main	Plot title
track	a dataframe of the cruise track (from the sds file) with longitude (long) and latitude (lat) named accordingly.
...	additional arguments to be passed to the plot function

Value

returns a map of seaflow statistics

Examples

```
## load the data
stat.tab <- system.file("extdata","seaflow_cruise","stats.tab",
package="flowPhyto")
stats <- read.delim(stat.tab)

## plot the cell concentrations of the picoplankton population
plotStatMap(df=stats ,pop='synecho', z.param='conc')
mtext(line=1, side=4, "cell concentration 10^6 cells / L")
```

readConsensusFile *Read a Consensus File.*

Description

Read a population classification consensus vector file from disk.

Usage

```
readConsensusFile(path)
```

Arguments

path	path to the consensus file

Value

a concensus vector in memory

Examples

```
opp.path <-  system.file("extdata","seaflow_cruise","2011_001", '1.evt.opp',
package="flowPhyto")

v <- readConsensusFile(opp.path)
table(v)
```

readPopDef *Read the Population Definition File.*

Description

Read the population definition file into memory from disk

Usage

```
readPopDef(pop.def.tab.path)
```

Arguments

pop.def.tab.path	Path to the population definition file or the cruise directory.

Value

a dataframe of population definition parameters

See Also

POP.DEF

Examples

```
seaflow.path <- system.file("extdata", 'seaflow_cruise', package="flowPhyto")
readPopDef(seaflow.path)
```

`readSeaflow` *Read a SeaFlow File*

Description

reads a binary SeaFlow event file and converts into an event dataframe

Usage

```
readSeaflow(file.path, column.names = EVT.HEADER, column.size = 2,
count.only=FALSE,  add.yearday.file=FALSE)
```

Arguments

`file.path`	System path to the binary seaflow event file.
`column.names`	Names of the channels. By default it uses the standard SeaFlow channels described in 'EVT.HEADER' that are [1] "time" [2] "pulse_width" [3] "D1" [4] "D2" [5] "fsc_small" [6] "fsc_perp" [7] "fsc_big" [8] "pe" [9] "chl_small" [9] "chl_big"
`column.size`	Sizes in bytes of the columns. Set up at 2 by default
`count.only`	Just read the first line of the file (the event/line count). Use to check the conversion of the binary file. FALSE by default
`add.yearday.file`	append the year_day and file number as two final columns in the returned dataframe.

Value

returns a seaflow dataframe

Examples

```
opp.file.path <- system.file("extdata","seaflow_cruise","2011_001", "1.evt.opp",
package="flowPhyto")

opp <- readSeaflow(opp.file.path)
```

summarize *Summarize phytoplankon event & log parameters*

Description

perform aggregate statistics on a particular combination of events (in a dataframe) per population.

Usage

```
summarize(x, channel.clmns = CHANNEL.CLMNS, opp.paths.str='1,2,3')
```

Arguments

x	event dataframe
channel.clmns	flow cyometry event-level chanels columns in the dataframe (x) on which to perform aggreate statistics (mean and standard deviation).
opp.paths.str	a comma delimited string of file names

Value

returns an aggregate statistics dataframe

Examples

```
opp.paths <- sapply(c(1,2,3), function(i)
      system.file("extdata","seaflow_cruise","2011_001", paste(i,'.evt.opp',sep=''),
          package="flowPhyto"))

filter.df <- do.call(rbind.data.frame, lapply(opp.paths, readSeaflow,
add.yearday.file=TRUE))
consen.df <- do.call(rbind.data.frame, lapply(opp.paths, readConsensusFile))
classed <- cbind.data.frame(filter.df, consen.df)


nrow.opp <- sapply(opp.paths, function(p) readSeaflow(              p , count.only=TRUE))
nrow.evt <- sapply(opp.paths, function(p) readSeaflow(sub('.opp','',p), count.only=TRUE))
classed$opp <- rep(nrow.opp, times=nrow.opp)
classed$evt <- rep(nrow.evt, times=nrow.opp)

summary(classed)

class.jn <- do.call(rbind.data.frame, by(classed, list(classed$file),
joinSDS, caroline::nv(opp.paths, sapply(opp.paths, getFileNumber)) ))
summarize(class.jn,  opp.paths.str=paste(names(opp.paths), collapse=','))
```

summarizeFile *Compute aggregate statistics on a collection of opp or fcs files.*

Description

perform aggregate statistics on a particular combination of filtered opp or fcs files for a particular population.

Usage

```
summarizeFile(opp.paths, pop.names, output.path=getCruisePath(opp.paths[1]))
```

Arguments

opp.paths	Path to the raw event file to be filtered.
pop.names	Abreviated name of the population subset to be summarized.
output.path	Path to the directory where you wish to output data.

Value

a single row dataframe (with header) file of per population aggregate statistics on both channels and log meta information

Examples

```
example.cruise.name <- 'seaflow_cruise'
temp.out.dir <- '.'

seaflow.path <- system.file("extdata", example.cruise.name, package="flowPhyto")
file.copy(from=seaflow.path, to=temp.out.dir, recursive=TRUE)

opp.paths <- sapply(c(1,2,3), function(i)
system.file("extdata","seaflow_cruise","2011_001", paste(i,'.evt.opp',sep=''),
package="flowPhyto"))


summarizeFile(opp.paths, pop.names='nano', output.path='.')

## optionally create a resample list
## to concatenate files conditionally on population concentrations
#resamp.list <- createResampleList(cruise.path,
# resample.min=500, resamp.concat.max=5)
unlink(example.cruise.name, recursive=TRUE)
```

validatePopDef *Validate a Population Definition Dataframe.*

Description

Validate the columns and values of the population definition dataframe passed to this function.

Usage

```
validatePopDef(pop.def)
```

Arguments

pop.def	Path to the raw event file to be filtered.

Value

a boolean indicating weither or not the pop def passed the validation check.

Examples

```
seaflow.path <- system.file("extdata", 'seaflow_cruise', package="flowPhyto")
pop.def <- readPopDef(seaflow.path)
validatePopDef(pop.def)
```

writeSeaflow *Write A SeaFlow File*

Description

writes a binary seaflow event file from a dataframe in memory

Usage

```
writeSeaflow(file.path, df, column.names = EVT.HEADER)
```

Arguments

file.path	System path to the binary seaflow event file.
df	SeaFlow dataframe in memory to be written to disk.
column.names	Names of the columns. By default it uses the global variable 'EVT.HEADER'

Examples

```
opp.path <- system.file("extdata","seaflow_cruise","2011_001", "1.evt.opp",
package="flowPhyto")
opp <- readSeaflow(opp.path)
writeSeaflow('./tmp.seaflow', opp)
Sys.sleep(30) # for windows build
```

Index