Package: iRfcb (via r-universe)

March 16, 2025

Type Package

Title Tools for Managing Imaging FlowCytobot (IFCB) Data

Version 0.4.3.9000

Description A comprehensive suite of tools for managing, processing, and analyzing data from the IFCB. I R FlowCytobot ('iRfcb') supports quality control, geospatial analysis, and preparation of IFCB data for publication in databases like <https://www.gbif.org>, <https://www.obis.org>, <https://emodnet.ec.europa.eu/en>, <https://shark.smhi.se/>, and <https://www.ecotaxa.org>. The package integrates with the MATLAB 'ifcb-analysis' tool, which is described in Sosik and Olson (2007) <doi:10.4319/lom.2007.5.204>, and provides features for working with raw, manually classified, and machine learning–classified image datasets. Key functionalities include image extraction, particle size distribution analysis, taxonomic data handling, and biomass concentration calculations, essential for plankton research.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports zip, lifecycle, shiny, stringr, dplyr, sf, reticulate (>= 1.41.0), tidyr, ggplot2, readr, worrms, png, R.matlab, curl, lubridate

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Note This package includes code from https://github.com/kudelalab/PSD by kudelalab licensed under the MIT License.

Suggests knitr, rmarkdown, testthat (>= 3.0.0), fs, mockery, shinytest2

VignetteBuilder knitr

Config/testthat/edition 3

URL https://europeanifcbgroup.github.io/iRfcb/,
https://github.com/EuropeanIFCBGroup/iRfcb

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev make libicu-dev libpng-dev libssl-dev libproj-dev python3 libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

Repository https://europeanifcbgroup.r-universe.dev

RemoteUrl https://github.com/europeanifcbgroup/irfcb

RemoteRef HEAD

RemoteSha 39618fc68c83fdc4b021bd87691f5aabb865c52b

Contents

`create_package_manifest`

Function to Create MANIFEST.txt

Description

This function generates a MANIFEST.txt file that lists all files in the specified paths, along with their sizes. It recursively includes files from directories and skips paths that do not exist. The manifest excludes the manifest file itself if present in the list.

Usage

```
create_package_manifest(paths, manifest_path = "MANIFEST.txt", temp_dir)
```

Arguments

`paths`	A character vector of paths to files and/or directories to include in the manifest.
`manifest_path`	A character string specifying the path to the manifest file. Default is "MANIFEST.txt".
`temp_dir`	A character string specifying the temporary directory to be removed from the file paths.

Value

This function does not return any value. It creates a `MANIFEST.txt` file at the specified location, which contains a list of all files (including their sizes) in the provided paths. The file paths are relative to the specified `temp_dir`, and the manifest excludes the manifest file itself if present.

`ifcb_adjust_classes` *Adjust Classifications in Manual Annotations*

Description

This function adjusts the classifications in manual annotation files based on a class2use file. It loads a specified class2use file and applies the adjustments to all relevant files in the specified manual folder. Optionally, it can also perform compression on the output files. This is the R equivalent function of `start_mc_adjust_classes_user_training` from the `ifcb-analysis repository` (Sosik and Olson 2007).

Usage

```
ifcb_adjust_classes(class2use_file, manual_folder, do_compression = TRUE)
```

Arguments

`class2use_file`	A character string representing the full path to the class2use file (should be a .mat file).
`manual_folder`	A character string representing the path to the folder containing manual annotation files. The function will look for files starting with 'D' in this folder.
`do_compression`	A logical value indicating whether to apply compression to the output files. Defaults to TRUE.

Details

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using `ifcb_py_install`.

Value

None

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

`ifcb_py_install` `ifcb_create_class2use` https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
ifcb_adjust_classes("C:/training/config/class2use", "C:/training/manual/2014/")

## End(Not run)
```

`ifcb_annotate_batch` *Annotate IFCB Images with Specified Class*

Description

This function creates or updates manual `.mat` classlist files with a user specified class in batch, based on input vector of IFCB image names. These `.mat` can be used with the code in the `ifcb-analysis` repository (Sosik and Olson 2007).

Usage

```
ifcb_annotate_batch(
  png_images,
  class,
  manual_folder,
  adc_folder,
  class2use_file,
  manual_output = NULL,
  manual_recursive = FALSE,
  unclassified_id = 1,
  do_compression = TRUE
)
```

Arguments

`png_images`	A character vector containing the names of the PNG images to be annotated in the format DYYYYMMDDTHHMMSS_IFCBXXX_ZZZZZ.png, where XXX represent the IFCB number and ZZZZZ the roi number.
`class`	A character string or integer specifying the class name or class2use index to annotate the images with. If a string is provided, it is matched against the available classes in `class2use_file`.
`manual_folder`	A character string specifying the path to the folder containing the manual `.mat` classlist files.
`adc_folder`	A character string specifying the path to the base folder containing the raw data, organized in subfolders by year (YYYY) and date (DYYYYMMDD). Each subfolder contains ADC files, which are used to determine the number of regions of interest (ROIs) for each sample when creating new manual `.mat` files.
`class2use_file`	A character string specifying the path to the `.mat` file containing class names and corresponding indices.
`manual_output`	A character string specifying the path to the folder where updated or newly created `.mat` classlist files will be saved. If not provided, the `manual_folder` path will be used by default.
`manual_recursive`	A logical value indicating whether to search recursively within `manual_folder` for `.mat` files. Default is `FALSE`.

`unclassified_id`
: An integer specifying the class ID to use for unclassified regions of interest (ROIs) when creating new manual `.mat` files. Default is `1`.

`do_compression`
: A logical value indicating whether to compress the .mat file. Default is TRUE.

Details

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using `ifcb_py_install`.

If an image belongs to a sample that already has a corresponding manual `.mat` file, the function updates the class IDs for the specified regions of interest (ROIs) in that file. If no manual file exists for the sample, the function creates a new one based on the sample's ADC data, assigning unclassified IDs to all ROIs initially, then applying the specified class to the relevant ROIs.

The class parameter can be provided as either a string (class name) or an integer (class index). If a string is provided, the function will attempt to match it to one of the available classes in `class2use_file`. If no match is found, an error is thrown.

The function assumes that the ADC files are organized in subfolders by year (YYYY) and date (DYYYYMMDD) within `adc_folder`.

Value

The function does not return a value. It creates or updates `.mat` files in the `manual_folder` to reflect the specified annotations.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

`ifcb_correct_annotation`, `ifcb_create_empty_manual_file`

Examples

```
## Not run:
# Initialize a python session if not already set up
ifcb_py_install()

# Annotate two png images with class "Nodularia_spumigena" and update or create manual files
ifcb_annotate_batch(
  png_images = c("D20230812T162908_IFCB134_01399.png",
                 "D20230714T102127_IFCB134_00069.png"),
  class = "Nodularia_spumigena",
  manual_folder = "path/to/manual",
  adc_folder = "path/to/adc",
  class2use_file = "path/to/class2use.mat"
)

## End(Not run)
```

`ifcb_convert_filenames` *Convert IFCB Filenames to Timestamps*

Description

This function converts IFCB filenames to a data frame with separate columns for the sample name, full timestamp, year, month, day, time, and IFCB number. ROI numbers are included if available.

Usage

```
ifcb_convert_filenames(filenames)
```

Arguments

`filenames`	A character vector of IFCB filenames in the format "DYYYYMMDDTHHMMSS_IFCBxxx".

Value

A tibble with columns:

- `sample`: The extracted sample name.
- `full_timestamp`: The full timestamp in "YYYY-MM-DD HH:MM:SS" format.
- `year`: The year as an integer.
- `month`: The month as an integer.
- `day`: The day as an integer.
- `time`: The extracted time in "HH:MM:SS" format.
- `ifcb_number`: The IFCB instrument number.

Examples

```
filenames <- c("D20230314T001205_IFCB134", "D20230615T123045_IFCB135")
timestamps <- ifcb_convert_filenames(filenames)
print(timestamps)
```

`ifcb_correct_annotation` *Correct Annotations in MATLAB Classlist Files*

Description

This function corrects annotations in MATLAB classlist files located in a specified manual folder, generated by the code in the `ifcb-analysis` repository (Sosik and Olson 2007). It replaces the class ID of specified regions of interest (ROIs) in the classlist files based on a correction file or a character vector.

Usage

```
ifcb_correct_annotation(
  manual_folder,
  out_folder,
  correction = NULL,
  correct_classid,
  do_compression = TRUE,
  correction_file = deprecated()
)
```

Arguments

`manual_folder`	A character string specifying the path to the folder containing the original MAT classlist files to be updated.
`out_folder`	A character string specifying the path to the folder where updated MAT classlist files will be saved.
`correction`	Either a character string specifying the path to the correction file, or a character vector containing image filenames to be corrected. If a file is provided, it should have a column named `image_filename`. If a character vector is provided, it will be treated as a direct list of image filenames.
`correct_classid`	An integer specifying the class ID to use for corrections.
`do_compression`	A logical value indicating whether to compress the .mat file. Default is TRUE.
`correction_file`	**[Deprecated]** Use `correction` instead.

Details

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using `ifcb_py_install`.

The correction file is expected to contain at least one column: `image_filename`, which includes the filenames of the images (with or without additional trailing information). The function processes each file, corrects the annotations, and saves the updated files in the output folder.

If a character vector is provided as `correction`, it will be used directly as a list of filenames for correction.

The `correction` is typically generated using a Shiny app that provides an interactive interface for browsing and managing IFCB (Imaging FlowCytobot) image galleries. This Shiny app can be initialized using the function `ifcb_run_image_gallery()`.

Value

This function does not return any value; it updates the classlist files in the specified output directory.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

`ifcb_py_install` https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Initialize a python session if not already set up
ifcb_py_install()

# Correct class ID in .mat classlist files using a correction file
ifcb_correct_annotation("input/manual",
                        "output/manual",
                        "corrections.txt",
                        99)

# Correct class ID in .mat classlist files using a character vector of filenames
ifcb_correct_annotation("input/manual",
                        "output/manual",
                        c("D20230917T153755_IFCB134_01724.png",
                          "D20230917T110059_IFCB134_00380.png"),
                        99)

## End(Not run)
```

---

ifcb_count_mat_annotations

Count IFCB Annotations from .mat Files

Description

This function processes .mat files, generated by the code in the `ifcb-analysis` repository (Sosik and Olson 2007), to count and summarize the annotations for each class based on the class2use information provided in a file.

Usage

```
ifcb_count_mat_annotations(
  manual_files,
  class2use_file,
  skip_class = NULL,
  sum_level = "class",
  mat_recursive = FALSE,
  use_python = FALSE
)
```

Arguments

manual_files	A character string specifying the path to the .mat files or a folder containing .mat files.
class2use_file	A character string specifying the path to the file containing the class2use variable.
skip_class	A numeric vector of class IDs or a character vector of class names to be excluded from the count. Default is NULL.
sum_level	A character string specifying the level of summarization. Options: "sample", "roi" or "class" (default).
mat_recursive	Logical. If TRUE, the function will search for MATLAB files recursively when manual_files is a folder. Default is FALSE.
use_python	Logical. If TRUE, attempts to read the .mat file using a Python-based method. Default is FALSE.

Details

If use_python = TRUE, the function tries to read the .mat file using ifcb_read_mat(), which relies on SciPy. This approach may be faster than the default approach using R.matlab::readMat(), especially for large .mat files. To enable this functionality, ensure Python is properly configured with the required dependencies. You can initialize the Python environment and install necessary packages using ifcb_py_install().

If use_python = FALSE or if SciPy is not available, the function falls back to using R.matlab::readMat().

Value

A data frame with the total count of images per class, roi or per sample.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

Examples

```
## Not run:
# Count annotations excluding specific class IDs
result <- ifcb_count_mat_annotations("path/to/manual_folder",
                                     "path/to/class2use_file",
                                     skip_class = c(99, 100))
print(result)

# Count annotations excluding a specific class name
result <- ifcb_count_mat_annotations("path/to/manual_folder",
                                     "path/to/class2use_file",
                                     skip_class = "unclassified")
print(result)

## End(Not run)
```

---

ifcb_create_class2use *Create a class2use .mat File*

Description

This function creates a `.mat` file containing a character vector of class names. A class2use file can be used for manual annotation using the code in the `ifcb-analysis` repository (Sosik and Olson 2007).

Usage

```
ifcb_create_class2use(classes, filename, do_compression = TRUE)
```

Arguments

`classes`	A character vector of class names to be saved in the `.mat` file.
`filename`	A string specifying the output file path (with `.mat` extension).
`do_compression`	A logical value indicating whether to compress the `.mat` file. Defaults to `TRUE`.

Details

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using `ifcb_py_install`.

Value

No return value. This function is called for its side effect of creating a `.mat` file.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

ifcb_py_install ifcb_adjust_classes https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Example usage:
classes <- c("unclassified", "Dinobryon_spp", "Helicostomella_spp")

ifcb_create_class2use(classes, "class2use_output.mat", do_compression = TRUE)

## End(Not run)
```

---

ifcb_create_empty_manual_file

Create an Empty Manual Classification MAT File

Description

Generates a MAT file for IFCB data with an empty manual classification structure using a specified number of ROIs, class names, and saves it to a specified output file. This function utilizes a Python script for creating the structure.

Usage

```
ifcb_create_empty_manual_file(
  roi_length,
  class2use,
  output_file,
  unclassified_id = 1,
  do_compression = TRUE
)
```

Arguments

roi_length	Integer. The number of rows in the class list (number of ROIs).
class2use	Character vector. The names of the classes to include in the class2use_manual field of the MAT file.
output_file	Character. The path where the output MAT file will be saved.
unclassified_id	Integer. The value to use in the second column of the class list. Default is 1.
do_compression	A logical value indicating whether to compress the .mat file. Default is TRUE.

Details

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using ifcb_py_install.

Value

No return value. This function is called for its side effects. The created MAT file is saved at the specified `output_file` location.

Examples

```
## Not run:
# Initialize a python session if not already set up
ifcb_py_install()

# Create a MAT file with 100 ROIs, using a vector of class names, and save it to "output.mat"
ifcb_create_empty_manual_file(roi_length = 100,
                              class2use = c("unclassified", "Aphanizomenon_spp"),
                              output_file = "output.mat")

# Create a MAT file with a different unclassified_id
ifcb_create_empty_manual_file(roi_length = 100,
                              class2use = c("unclassified", "Aphanizomenon_spp"),
                              output_file = "output.mat",
                              unclassified_id = 999)

## End(Not run)
```

---

ifcb_create_manifest *Create a MANIFEST.txt File*

Description

This function generates a MANIFEST.txt file listing all files in a specified folder and its subfolders, along with their sizes in bytes. The function can optionally exclude an existing MANIFEST.txt file from the generated list. A manifest may be useful when archiving images in data repositories.

Usage

```
ifcb_create_manifest(
  folder_path,
  manifest_path = file.path(folder_path, "MANIFEST.txt"),
  exclude_manifest = TRUE
)
```

Arguments

`folder_path`	A character string specifying the path to the folder whose files are to be listed.
`manifest_path`	A character string specifying the path and name of the MANIFEST.txt file to be created. Defaults to "folder_path/MANIFEST.txt".
`exclude_manifest`	A logical value indicating whether to exclude an existing MANIFEST.txt file from the list. Defaults to TRUE.

Value

No return value, called for side effects. Creates a MANIFEST.txt file at the specified location.

Examples

```
## Not run:
# Create a MANIFEST.txt file for the current directory
ifcb_create_manifest(".")

# Create a MANIFEST.txt file for a specific directory, excluding an existing MANIFEST.txt file
ifcb_create_manifest("path/to/directory")

# Create a MANIFEST.txt file and save it to a specific path
ifcb_create_manifest("path/to/directory", manifest_path = "path/to/manifest/MANIFEST.txt")

# Create a MANIFEST.txt file without excluding an existing MANIFEST.txt file
ifcb_create_manifest("path/to/directory", exclude_manifest = FALSE)

## End(Not run)
```

---

ifcb_download_test_data

Download Test IFCB Data

Description

This function downloads a zip archive containing MATLAB files from the iRfcb dataset available in the SMHI IFCB Plankton Image Reference Library (Torstensson et al. 2024), unzips them into the specified folder and extracts png images. These data can be used, for instance, for testing iRfcb and for creating the tutorial vignette using `vignette("tutorial", package = "iRfcb")`

Usage

```
ifcb_download_test_data(
  dest_dir,
  figshare_article = "48158716",
  max_retries = 5,
  sleep_time = 10,
  verbose = TRUE
)
```

Arguments

dest_dir	The destination directory where the files will be unzipped.
figshare_article	The file article number at the SciLifeLab Figshare data repository. By default, the iRfcb test dataset (48158716) from Torstensson et al. (2024) is used.

max_retries	The maximum number of retry attempts in case of download failure. Default is 5.
sleep_time	The sleep time between download attempts, in seconds. Default is 10.
verbose	A logical indicating whether to print progress messages. Default is TRUE.

Value

No return value. This function is called for its side effect of downloading, extracting, and organizing IFCB test data.

References

Torstensson, Anders; Skjevik, Ann-Turi; Mohlin, Malin; Karlberg, Maria; Karlson, Bengt (2024). SMHI IFCB Plankton Image Reference Library. Version 3. SciLifeLab. Dataset. doi:10.17044/scilifelab.25883455.v3

Examples

```
## Not run:
# Download and unzip IFCB test data into the "data" directory
ifcb_download_test_data("data")

## End(Not run)
```

---

ifcb_extract_annotated_images

Extract Annotated Images from IFCB Data

Description

This function extracts labeled images from IFCB (Imaging FlowCytobot) data, annotated using the MATLAB code from the `ifcb-analysis` repository (Sosik and Olson 2007). It reads manually classified data, maps class indices to class names, and extracts the corresponding Region of Interest (ROI) images, saving them to the specified directory.

Usage

```
ifcb_extract_annotated_images(
  manual_folder,
  class2use_file,
  roi_folders,
  out_folder,
  skip_class = NA,
  verbose = TRUE,
  manual_recursive = FALSE,
  roi_recursive = TRUE,
```

```
  overwrite = FALSE,
  use_python = FALSE,
  roi_folder = deprecated()
)
```

Arguments

manual_folder	A character string specifying the path to the directory containing the manually classified .mat files.
class2use_file	A character string specifying the path to the file containing class names.
roi_folders	A character vector specifying one or more directories containing the ROI files.
out_folder	A character string specifying the output directory where the extracted images will be saved.
skip_class	A numeric vector of class IDs or a character vector of class names to be excluded from the count. Default is NULL.
verbose	A logical value indicating whether to print progress messages. Default is TRUE.
manual_recursive	Logical. If TRUE, the function will search for MATLAB files recursively within the manual_folder. Default is FALSE.
roi_recursive	Logical. If TRUE, the function will search for data files recursively within the roi_folder (if provided). Default is TRUE.
overwrite	A logical value indicating whether to overwrite existing PNG files. Default is FALSE.
use_python	Logical. If TRUE, attempts to read the .mat file using a Python-based method. Default is FALSE.
roi_folder	**[Deprecated]** Use roi_folders instead.

Details

If use_python = TRUE, the function tries to read the .mat file using ifcb_read_mat(), which relies on SciPy. This approach may be fasterthan the default approach using R.matlab::readMat(), especially for large .mat files. To enable this functionality, ensure Python is properly configured with the required dependencies. You can initialize the Python environment and install necessary packages using ifcb_py_install().

If use_python = FALSE or if SciPy is not available, the function falls back to using R.matlab::readMat().

Value

None. The function saves the extracted PNG images to the specified output directory.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

ifcb_extract_pngs ifcb_extract_classified_images https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
ifcb_extract_annotated_images(
  manual_folder = "path/to/manual_folder",
  class2use_file = "path/to/class2use_file.mat",
  roi_folders = "path/to/roi_folder",
  out_folder = "path/to/out_folder",
  skip_class = 1
)

## End(Not run)
```

---

ifcb_extract_biovolumes

Extract Biovolumes from IFCB Data and Compute Carbon Content

Description

This function reads biovolume data from feature files generated by the `ifcb-analysis` repository (Sosik and Olson 2007) and matches them with corresponding classification results or manual annotations. It calculates biovolume in cubic micrometers and determines if each class is a diatom based on the World Register of Marine Species (WoRMS). Carbon content is computed for each region of interest (ROI) using conversion functions from Menden-Deuer and Lessard (2000), depending on whether the class is identified as a diatom.

Usage

```
ifcb_extract_biovolumes(
  feature_files,
  mat_folder = NULL,
  custom_images = NULL,
  custom_classes = NULL,
  class2use_file = NULL,
  micron_factor = 1/3.4,
  diatom_class = "Bacillariophyceae",
  marine_only = FALSE,
  threshold = "opt",
  multiblob = FALSE,
  feature_recursive = TRUE,
  mat_recursive = TRUE,
  use_python = FALSE,
  verbose = TRUE
)
```

Arguments

feature_files	A path to a folder containing feature files or a character vector of file paths.
mat_folder	(Optional) Path to the folder containing class or manual annotation files.
custom_images	(Optional) A character vector of image filenames in the format DYYYYMMDDTHHMMSS_IFCBXXX_ZZZZZ.png, where "XXX" represents the IFCB number and "ZZZZZ" represents the ROI number. These filenames should match the roi_number assignment in the feature_files and can be used as a substitute for MATLAB files.
custom_classes	(Optional) A character vector of corresponding class labels for custom_images.
class2use_file	A character string specifying the path to the file containing the class2use variable (default: NULL).
micron_factor	Conversion factor for biovolume to cubic micrometers. Default is 1 / 3.4.
diatom_class	A character vector specifying diatom class names in WoRMS. Default: "Bacillariophyceae".
marine_only	Logical. If TRUE, restricts the WoRMS search to marine taxa only. Default: FALSE.
threshold	Threshold for selecting classification information ("opt" for above-threshold classification, otherwise "all"). Default: "opt".
multiblob	Logical. If TRUE, includes multiblob features. Default: FALSE.
feature_recursive	Logical. If TRUE, searches recursively for feature files when feature_files is a folder. Default: TRUE.
mat_recursive	Logical. If TRUE, searches recursively for MATLAB files in mat_folder. Default: TRUE.
use_python	Logical. If TRUE, attempts to read .mat files using a Python-based method (SciPy). Default: FALSE.
verbose	Logical. If TRUE, prints progress messages. Default: TRUE.

Details

- **Classification Data Handling:**
  - If mat_folder is provided, the function reads class annotations from MATLAB .mat files.
  - If custom_images and custom_classes are supplied, they override MATLAB classification data (e.g. data from a CNN model).
  - If both mat_folder and custom_images/custom_classes are given, mat_folder takes precedence.
- **MAT File Processing:**
  - If use_python = TRUE, the function reads .mat files using ifcb_read_mat() (requires Python + SciPy).
  - Otherwise, it falls back to R.matlab::readMat().

Value

A data frame containing:

- `sample`: The sample name.
- `classifier`: The classifier used (if applicable).
- `roi_number`: The region of interest (ROI) number.
- `class`: The identified taxonomic class.
- `biovolume_um3`: Computed biovolume in cubic micrometers.
- `carbon_pg`: Estimated carbon content in picograms.

References

Menden-Deuer Susanne, Lessard Evelyn J., (2000), Carbon to volume relationships for dinoflagellates, diatoms, and other protist plankton, Limnology and Oceanography, 3, doi: 10.4319/lo.2000.45.3.0569.

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

`ifcb_read_features` `ifcb_is_diatom` https://www.marinespecies.org/

Examples

```
## Not run:
# Using MATLAB results:
feature_files <- "data/features"
mat_folder <- "data/classified"
biovolume_df <- ifcb_extract_biovolumes(feature_files,
                                        mat_folder)
print(biovolume_df)

# Using custom classification result:
class = c("Mesodinium_rubrum",
          "Mesodinium_rubrum")
image <- c("D20220522T003051_IFCB134_00002",
           "D20220522T003051_IFCB134_00003")
biovolume_df_custom <- ifcb_extract_biovolumes(feature_files,
                                        custom_images = image,
                                        custom_classes = class)
print(biovolume_df_custom)

## End(Not run)
```

`ifcb_extract_classified_images` *Extract Taxa Images from MATLAB Classified Sample*

Description

This function reads a MATLAB classified sample file (.mat) generated by the `start_classify_batch_user_training` function from the `ifcb-analysis` repository (Sosik and Olson 2007), extracts specified taxa images from the corresponding ROI files, and saves each image in a specified directory.

Usage

```
ifcb_extract_classified_images(
  sample,
  classified_folder,
  roi_folder,
  out_folder,
  taxa = "All",
  threshold = "opt",
  overwrite = FALSE,
  use_python = FALSE,
  verbose = TRUE
)
```

Arguments

`sample`	A character string specifying the sample name.
`classified_folder`	A character string specifying the directory containing the classified files.
`roi_folder`	A character string specifying the directory containing the ROI files.
`out_folder`	A character string specifying the directory to save the extracted images.
`taxa`	A character string specifying the taxa to extract. Default is "All".
`threshold`	A character string specifying the threshold to use ("none", "opt", "adhoc"). Default is "opt".
`overwrite`	A logical value indicating whether to overwrite existing PNG files. Default is FALSE.
`use_python`	Logical. If `TRUE`, attempts to read the `.mat` file using a Python-based method. Default is `FALSE`.
`verbose`	A logical value indicating whether to print progress messages. Default is TRUE.

Details

If `use_python = TRUE`, the function tries to read the `.mat` file using `ifcb_read_mat()`, which relies on `SciPy`. This approach may be faster than the default approach using `R.matlab::readMat()`, especially for large `.mat` files. To enable this functionality, ensure Python is properly configured

with the required dependencies. You can initialize the Python environment and install necessary packages using `ifcb_py_install()`.

If `use_python = FALSE` or if `SciPy` is not available, the function falls back to using `R.matlab::readMat()`.

Value

No return value, called for side effects. Extracts and saves taxa images to a directory.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

`ifcb_extract_pngs` `ifcb_extract_annotated_images` https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Define the parameters
sample <- "D20230311T092911_IFCB135"
classified_folder <- "path/to/classified_folder"
roi_folder <- "path/to/roi_folder"
out_folder <- "path/to/outputdir"
taxa <- "All"  # or specify a particular taxa
threshold <- "opt"  # or specify another threshold

# Extract taxa images from the classified sample
ifcb_extract_classified_images(sample, classified_folder, roi_folder, out_folder, taxa, threshold)

## End(Not run)
```

---

ifcb_extract_pngs *Extract Images from IFCB ROI File*

Description

This function reads an IFCB (.roi) file and its corresponding .adc file, extracts regions of interest (ROIs), and saves each ROI as a PNG image in a specified directory. Optionally, you can specify ROI numbers to extract, useful for specific ROIs from manual or automatic classification results.

Usage

```
ifcb_extract_pngs(
  roi_file,
  out_folder = dirname(roi_file),
  ROInumbers = NULL,
  taxaname = NULL,
```

```
  gamma = 1,
  verbose = TRUE,
  overwrite = FALSE
)
```

Arguments

roi_file	A character string specifying the path to the .roi file.
out_folder	A character string specifying the directory where the PNG images will be saved. Defaults to the directory of the ROI file.
ROInumbers	An optional numeric vector specifying the ROI numbers to extract. If NULL, all ROIs with valid dimensions are extracted.
taxaname	An optional character string specifying the taxa name for organizing images into subdirectories. Defaults to NULL.
gamma	A numeric value for gamma correction applied to the image. Default is 1 (no correction). Values <1 increase contrast in dark regions, while values >1 decrease contrast.
verbose	A logical value indicating whether to print progress messages. Default is TRUE.
overwrite	A logical value indicating whether to overwrite existing PNG files. Default is FALSE.

Value

This function is called for its side effects: it writes PNG images to a directory.

See Also

ifcb_extract_classified_images for extracting ROIs from automatic classification.

ifcb_extract_annotated_images for extracting ROIs from manual annotation.

Examples

```
## Not run:
# Convert ROI file to PNG images
ifcb_extract_pngs("path/to/your_roi_file.roi")

# Extract specific ROI numbers from ROI file
ifcb_extract_pngs("path/to/your_roi_file.roi", "output_directory", ROInumbers = c(1, 2, 3))

## End(Not run)
```

`ifcb_get_ecotaxa_example` *Get Ecotaxa Column Names*

Description

This function reads an example EcoTaxa metadata file included in the `iRfcb` package.

Usage

```
ifcb_get_ecotaxa_example(example = "ifcb")
```

Arguments

`example`	A character string specifying which example EcoTaxa metadata file to load. Options are: **"minimal"** Loads a minimal example, for fully manual entry. **"full_unknown"** Loads a full featured example, with unknown objects only. **"full_classified"** Loads a full featured example, with already classified objects. **"ifcb"** (Default) Loads a full IFCB-specific dataset used for EcoTaxa submissions.

Details

This function loads different types of EcoTaxa metadata examples based on the user's need. The examples include a minimal template for manual data entry, as well as fully featured datasets with or without classified objects. The default is an IFCB-specific example, originating from https://github.com/VirginieSonnet/IFCBdatabaseToEcotaxa. The example headers can used when submitting data from Imaging FlowCytobot (IFCB) instruments to EcoTaxa at https://ecotaxa.obs-vlfr.fr/.

Value

A data frame containing EcoTaxa example metadata.

Examples

```
ecotaxa_example <- ifcb_get_ecotaxa_example()

# Print the first five columns
dplyr::tibble(ecotaxa_example)
```

`ifcb_get_ferrybox_data` *Retrieve Ferrybox Data for Specified Timestamps*

Description

This internal SMHI function reads `.txt` files from a specified folder containing Ferrybox data, filters them based on a specified ship name (default is "SveaFB" for R/V Svea), and extracts data (including GPS coordinates) for timestamps (rounded to the nearest minute) falling within the date ranges defined in the file names.

Usage

```
ifcb_get_ferrybox_data(
  timestamps,
  ferrybox_folder,
  parameters = c("8002", "8003"),
  ship = "SveaFB",
  latitude_param = "8002",
  longitude_param = "8003"
)
```

Arguments

`timestamps`	A vector of POSIXct timestamps for which GPS coordinates and associated parameter data are to be retrieved.
`ferrybox_folder`	A string representing the path to the folder containing Ferrybox `.txt` files.
`parameters`	A character vector specifying the parameters to extract from the Ferrybox data. Defaults to `c("8002", "8003")`.
`ship`	A string representing the name of the ship to filter Ferrybox files. The default is "SveaFB".
`latitude_param`	A string specifying the header name for the latitude column in the Ferrybox data. Default is "8002".
`longitude_param`	A string specifying the header name for the longitude column in the Ferrybox data. Default is "8003".

Details

The function extracts data from files whose names match the specified ship and fall within the date ranges defined in the file names. The columns corresponding to `latitude_param` and `longitude_param` will be renamed to `gpsLatitude` and `gpsLongitude`, respectively, if they are present in the `parameters` argument.

The function also handles cases where the exact timestamp is missing by attempting to interpolate the data using floor and ceiling rounding methods. The final output will ensure that all specified parameters are numeric.

Value

A data frame containing the input timestamps and corresponding data for the specified parameters. Columns include 'timestamp', 'gpsLatitude', 'gpsLongitude' (if applicable), and the specified parameters.

Examples

```
## Not run:
ferrybox_folder <- "/path/to/ferrybox/data"
timestamps <- as.POSIXct(c("2016-08-10 10:47:34 UTC",
                           "2016-08-10 11:12:21 UTC",
                           "2016-08-10 11:35:59 UTC"))

result <- ifcb_get_ferrybox_data(timestamps, ferrybox_folder)
print(result)

## End(Not run)
```

---

ifcb_get_mat_names *Get Variable Names from a MAT File*

Description

This function reads a .mat file generated the `ifcb-analysis` repository (Sosik and Olson 2007) and retrieves the names of all variables stored within it.

Usage

```
ifcb_get_mat_names(mat_file, use_python = FALSE)
```

Arguments

`mat_file`	A character string specifying the path to the .mat file.
`use_python`	Logical. If `TRUE`, attempts to read the `.mat` file using a Python-based method. Default is `FALSE`.

Details

If `use_python = TRUE`, the function tries to read the `.mat` file using `ifcb_read_mat()`, which relies on `SciPy`. This approach may be faster than the default approach using `R.matlab::readMat()`, especially for large `.mat` files. To enable this functionality, ensure Python is properly configured with the required dependencies. You can initialize the Python environment and install necessary packages using `ifcb_py_install()`.

If `use_python = FALSE` or if `SciPy` is not available, the function falls back to using `R.matlab::readMat()`.

Value

A character vector of variable names.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

ifcb_get_mat_variable https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Get variable names from a MAT file
variables <- ifcb_get_mat_names("path/to/file.mat")
print(variables)

## End(Not run)
```

---

ifcb_get_mat_variable *Get Classes from a MAT File*

Description

This function reads a specified variable from a .mat file generated by the `ifcb-analysis` repository (Sosik and Olson 2007). It can be used, for example, to extract lists of classes from the file.

Usage

```
ifcb_get_mat_variable(
  mat_file,
  variable_name = "class2use",
  use_python = FALSE
)
```

Arguments

`mat_file`	A character string specifying the path to the .mat file containing the class information.
`variable_name`	A character string specifying the variable name in the .mat file that contains the class information. The default is "class2use". Other examples include "class2use_manual" from a manual file, or "class2use_auto" for a class list used for automatic assignment. You can find available variable names using the function `ifcb_get_mat_names`.
`use_python`	Logical. If `TRUE`, attempts to read the `.mat` file using a Python-based method. Default is `FALSE`.

Details

If use_python = TRUE, the function tries to read the .mat file using ifcb_read_mat(), which relies on SciPy. This approach may be faster than the default approach using R.matlab::readMat(), especially for large .mat files. To enable this functionality, ensure Python is properly configured with the required dependencies. You can initialize the Python environment and install necessary packages using ifcb_py_install().

If use_python = FALSE or if SciPy is not available, the function falls back to using R.matlab::readMat().

Value

A character vector of class names.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

ifcb_get_mat_names https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Get class names from a class2use file
classes <- ifcb_get_mat_variable("path/to/class2use.mat", "class2use")
print(classes)

# Get class names from a classifier file
class2useTB <- ifcb_get_mat_variable("path/to/classified/sample.mat", "class2useTB")
print(class2useTB)

## End(Not run)
```

---

ifcb_get_runtime *Read IFCB Header File and Extract Runtime Information*

Description

This function imports an IFCB header file (either from a local path or URL), extracts specific target values such as runtime and inhibittime, and returns them in a structured format (in seconds). This is the R equivalent function of IFCBxxx_readhdr from the ifcb-analysis repository (Sosik and Olson 2007).

Usage

```
ifcb_get_runtime(hdr_file)
```

Arguments

hdr_file	A character string specifying the full path to the .hdr file or URL.

Value

A list (hdr) containing runtime, inhibittime, and runType (if available) extracted from the header file.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Example: Read and extract information from an IFCB header file
hdr_info <- ifcb_get_runtime("path/to/IFCB_hdr_file.hdr")
print(hdr_info)

## End(Not run)
```

---

ifcb_get_shark_colnames

Get Shark Column Names

Description

This function reads SHARK column names from a specified tab-separated values (TSV) file included in the package. These columns are used for submitting IFCB data to https://shark.smhi.se/.

Usage

```
ifcb_get_shark_colnames(minimal = FALSE)
```

Arguments

minimal	A logical value indicating whether to load only the minimal set of column names required for data submission to SHARK. Default is FALSE.

Details

For a detailed example of a data submission, see ifcb_get_shark_example.

Value

An empty data frame containing the SHARK column names.

See Also

ifcb_get_shark_example

Examples

```
shark_colnames <- ifcb_get_shark_colnames()
print(shark_colnames)

shark_colnames_minimal <- ifcb_get_shark_colnames(minimal = TRUE)
print(shark_colnames_minimal)
```

---

ifcb_get_shark_example

Get Shark Column Example

Description

This function reads a SHARK submission example from a file included in the package. This format is used for submitting IFCB data to https://shark.smhi.se/.

Usage

```
ifcb_get_shark_example()
```

Value

A data frame containing example data following the SHARK submission format.

See Also

ifcb_get_shark_colnames

Examples

```
shark_example <- ifcb_get_shark_example()

# Print example as tibble
dplyr::tibble(shark_example)
```

`ifcb_get_trophic_type` *Get Trophic Type for a List of Plankton Taxa*

Description

This function matches a specified list of taxa with a summarized list of trophic types for various plankton taxa from Northern Europe (data sourced from `SMHI Trophic Type`).

Usage

```
ifcb_get_trophic_type(taxa_list = NULL, print_complete_list = FALSE)
```

Arguments

`taxa_list`	A character vector of scientific names for which trophic types are to be retrieved.
`print_complete_list`	Logical, if TRUE, prints the complete list of summarized trophic types.

Details

If there are multiple trophic types for a scientific name (i.e. AU and HT size classes), the summarized trophic type is "NS".

Value

A character vector of trophic types corresponding to the scientific names in `taxa_list`, or a data frame containing all taxa and trophic types available in the `SMHI Trophic Type` list. The available trophic types are autotrophic (AU), heterotrophic (HT), mixotrophic (MX) or not specified (NS).

Examples

```
# Example usage:
taxa_list <- c("Acanthoceras zachariasii",
               "Nodularia spumigena",
               "Acanthoica quattrospina",
               "Noctiluca",
               "Gymnodiniales")

ifcb_get_trophic_type(taxa_list)
```

`ifcb_is_diatom` *Identify Diatoms in Taxa List*

Description

This function takes a list of taxa names, cleans them, retrieves their corresponding classification records from the World Register of Marine Species (WoRMS), and checks if they belong to the specified diatom class. The function only uses the first name (genus name) of each taxa for classification.

Usage

```
ifcb_is_diatom(
  taxa_list,
  diatom_class = "Bacillariophyceae",
  max_retries = 3,
  sleep_time = 10,
  marine_only = FALSE,
  fuzzy = deprecated(),
  verbose = TRUE
)
```

Arguments

`taxa_list`	A character vector containing the list of taxa names.
`diatom_class`	A character string or vector specifying the class name(s) to be identified as diatoms, according to WoRMS. Default is "Bacillariophyceae".
`max_retries`	An integer specifying the maximum number of attempts to retrieve WoRMS records in case of an error. Default is 3.
`sleep_time`	A numeric value indicating the number of seconds to wait between retry attempts. Default is 10 seconds.
`marine_only`	Logical. If TRUE, restricts the search to marine taxa only. Default is FALSE.
`fuzzy`	**[Deprecated]** The fuzzy argument is no longer available
`verbose`	A logical indicating whether to print progress messages. Default is TRUE.

Value

A logical vector indicating whether each cleaned taxa name belongs to the specified diatom class.

See Also

https://www.marinespecies.org/

Examples

```
## Not run:
taxa_list <- c("Nitzschia_sp", "Chaetoceros_sp", "Dinophysis_norvegica", "Thalassiosira_sp")
ifcb_is_diatom(taxa_list)

## End(Not run)
```

`ifcb_is_in_basin` *Check if Points are in a Specific Sea Basin*

Description

This function checks if vectors of latitude and longitude points are within a user-supplied sea basin. The Baltic Sea basins are included as a pre-packaged shapefile in the `iRfcb` package.

Usage

```
ifcb_is_in_basin(latitudes, longitudes, plot = FALSE, shape_file = NULL)
```

Arguments

`latitudes`	A numeric vector of latitude points.
`longitudes`	A numeric vector of longitude points.
`plot`	A boolean indicating whether to plot the points and the sea basin. Default is FALSE.
`shape_file`	The absolute path to a custom polygon shapefile in WGS84 (EPSG:4326) that represents the specific sea basin. Default is a land-buffered shapefile of the Baltic Sea basins, included in the `iRfcb` package.

Details

This function reads a pre-packaged shapefile of the Baltic Sea Basin from the `iRfcb` package by default, or a user-supplied shapefile if provided. It sets the CRS, transforms the CRS to WGS84 (EPSG:4326) if necessary, and checks if the given points fall within the specified sea basin. Optionally, it plots the points and the sea basin polygons together.

Value

A logical vector indicating whether each point is within the specified sea basin, or a plot with the points and basins if `plot = TRUE`.

Examples

```
# Define example latitude and longitude vectors
latitudes <- c(55.337, 54.729, 56.311, 57.975)
longitudes <- c(12.674, 14.643, 12.237, 10.637)

# Check if the points are in the Baltic Sea Basin
points_in_the_baltic <- ifcb_is_in_basin(latitudes, longitudes)
print(points_in_the_baltic)

# Plot the points and the basin
ifcb_is_in_basin(latitudes, longitudes, plot = TRUE)
```

---

ifcb_is_near_land *Determine if Positions are Near Land*

Description

Determines whether given positions are near land based on a coastline shape file. The Natural Earth 1:10m land vectors are included as default shapefile in `iRfcb`.

Usage

```
ifcb_is_near_land(
  latitudes,
  longitudes,
  distance = 500,
  shape = NULL,
  crs = 4326,
  utm_zone = 33,
  remove_small_islands = TRUE,
  small_island_threshold = 2e+06
)
```

Arguments

`latitudes`	Numeric vector of latitudes for positions.
`longitudes`	Numeric vector of longitudes for positions.
`distance`	Buffer distance in meters around the coastline. Default is 500 m.
`shape`	Optional path to a shapefile containing coastline data. If provided, the function will use this shapefile instead of the default Natural Earth 1:10m land vectors. Using a more detailed shapefile allows for a smaller buffer distance. For detailed European coastlines, download polygons from the EEA at https://www.eea.europa.eu/data-and-maps/data/eea-coastline-for-analysis-2/gis-data/eea-coastline-polygon. For more detailed world maps, download from Natural Earth at https://www.naturalearthdata.com/downloads/10m-physical-vectors/.

crs Coordinate reference system (CRS) to use for positions and output. Default is EPSG code 4326 (WGS84).

utm_zone UTM zone for buffering the coastline. Default is 33 (between 12°E and 18°E, northern hemisphere).

remove_small_islands
Logical indicating whether to remove small islands from the coastline if a custom shapefile is provided. Default is TRUE.

small_island_threshold
Area threshold in square meters below which islands will be considered small and removed, if remove_small_islands is set to TRUE. Default is 2 sqkm.

Details

This function calculates a buffered area around the coastline and checks if given positions (specified by longitudes and latitudes) are within this buffer or intersect with land.

Value

Logical vector indicating whether each position is near land.

Examples

```
# Define coordinates
latitudes <- c(62.500353, 58.964498, 57.638725, 56.575338)
longitudes <- c(17.845993, 20.394418, 18.284523, 16.227174)

# Call the function
near_land <- ifcb_is_near_land(latitudes, longitudes, distance = 300, crs = 4326)

# Print the result
print(near_land)
```

---

ifcb_match_taxa_names *Retrieve WoRMS Records with Retry Mechanism*

Description

This function attempts to retrieve WoRMS records using the provided taxa names. It retries the operation if an error occurs, up to a specified number of attempts.

Usage

```
ifcb_match_taxa_names(
  taxa_names,
  best_match_only = TRUE,
  max_retries = 3,
  sleep_time = 10,
```

```
  marine_only = FALSE,
  return_list = FALSE,
  verbose = TRUE,
  fuzzy = deprecated()
)
```

Arguments

taxa_names	A character vector of taxa names to retrieve records for.
best_match_only	A logical value indicating whether to automatically select the first match and return a single match. Default is TRUE.
max_retries	An integer specifying the maximum number of attempts to retrieve records.
sleep_time	A numeric value indicating the number of seconds to wait between retry attempts.
marine_only	Logical. If TRUE, restricts the search to marine taxa only. Default is FALSE.
return_list	A logical value indicating whether to to return the output as a list. Default is FALSE, where the result is returned as a dataframe.
verbose	A logical indicating whether to print progress messages. Default is TRUE.
fuzzy	**[Deprecated]** The fuzzy argument is no longer available

Value

A data frame (or list if return_list is TRUE) of WoRMS records or NULL if the retrieval fails after the maximum number of attempts.

Examples

```
## Not run:
# Example: Retrieve WoRMS records for a list of taxa names
taxa <- c("Calanus finmarchicus", "Thalassiosira pseudonana", "Phaeodactylum tricornutum")
records <- ifcb_match_taxa_names(taxa_names = taxa,
                                 max_retries = 3,
                                 sleep_time = 5,
                                 marine_only = TRUE,
                                 verbose = TRUE)

print(records)

## End(Not run)
```

`ifcb_merge_manual` *Merge IFCB Manual Classification Data*

Description

This function merges two sets of manual classification data by combining and aligning class labels from a base set and an additional set of classifications. The merged `.mat` data can be used with the code in the `ifcb-analysis` repository (Sosik and Olson 2007).

Usage

```
ifcb_merge_manual(
  class2use_file_base,
  class2use_file_additions,
  class2use_file_output = NULL,
  manual_folder_base,
  manual_folder_additions,
  manual_folder_output,
  do_compression = TRUE,
  temp_index_offset = 50000,
  skip_class = NULL,
  quiet = FALSE
)
```

Arguments

`class2use_file_base`
: Character. Path to the `class2use` file of the base manual classifications. The base set contains the original manual classifications list that form the foundation for merging.

`class2use_file_additions`
: Character. Path to the `class2use` file of the additions manual classifications. The additions set contains additional classifications that need to be merged with the base set. Class labels from the `class2use_file_additions` that are not already included in the `class2use_file_base` will be added to generate the `class2use_file_output`.

`class2use_file_output`
: Character. Path where the merged `class2use` file will be saved. If `NULL`, the merged file will be stored in the same directory as `class2use_file_base`. Default is `NULL`.

`manual_folder_base`
: Character. Path to the folder containing the base set of manual classification `.mat` files.

`manual_folder_additions`
: Character. Path to the folder containing the additions set of manual classification `.mat` files.

`manual_folder_output`
: Character. Path to the output folder where the merged classification files will be stored.

`do_compression`
: A logical value indicating whether to compress the `.mat` file. Defaults to `TRUE`.

`temp_index_offset`
: Numeric. A large integer used to generate temporary indices during the merge process. Default is 50000.

`skip_class`
: Character. A vector of class names to skip from the `class2use_file_additions` during the merge process. Default is `NULL`.

`quiet`
: Logical. If `TRUE`, suppresses output messages. Default is `FALSE`.

Details

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using `ifcb_py_install`.

The **base** set consists of the original classifications that are used as a reference for the merging process. The **additions** set contains the additional classifications that need to be merged with the base set. When merging, unique class names from the additions set that are not present in the base set are appended.

The function works by aligning the class labels from the additions set with those in the base set, handling conflicts by using a temporary index system. It copies `.mat` files from both the base and additions folders into the output folder, while adjusting indices and and class names for the additions.

Note that the maximum limit for `uint16` is 65,535, so ensure that `temp_index_offset` remains below this value.

Value

No return value. Outputs the combined `class2use` file in the same folder as `class2use_file_base` is located or at a user-specified location, and merged `.mat` files into the output folder.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

`ifcb_py_install` https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
ifcb_merge_manual("path/to/class2use_base.mat", "path/to/class2use_additions.mat",
                  "path/to/class2use_combined.mat", "path/to/manual/base_folder",
                  "path/to/manual/additions_folder", "path/to/manual/output_folder",
                  do_compression = TRUE, temp_index_offset = 50000, quiet = FALSE)
```

```
## End(Not run)
```

`ifcb_psd` *Plot and Save IFCB PSD Data*

Description

This function generates and saves data about a dataset's Particle Size Distribution (PSD) from Imaging FlowCytobot (IFCB) feature and hdr files, which can be used for data quality assurance and quality control.

Usage

```
ifcb_psd(
  feature_folder,
  hdr_folder,
  save_data = FALSE,
  output_file = NULL,
  plot_folder = NULL,
  use_marker = FALSE,
  start_fit = 10,
  r_sqr = 0.5,
  beads = NULL,
  bubbles = NULL,
  incomplete = NULL,
  missing_cells = NULL,
  biomass = NULL,
  bloom = NULL,
  humidity = NULL,
  micron_factor = 1/3.4
)
```

Arguments

`feature_folder`	The absolute path to a directory containing all of the v2 feature files for the dataset.
`hdr_folder`	The absolute path to a directory containing all of the hdr files for the dataset.
`save_data`	A boolean indicating whether to save data to CSV files. Default is FALSE.
`output_file`	A string with the base file name for the .csv to use (including path). Set to NULL to not save data (default).
`plot_folder`	The folder where graph images for each file will be saved. Set to NULL to not save graphs (default).
`use_marker`	A boolean indicating whether to show markers on the plot. Default is FALSE.
`start_fit`	An integer indicating the start fit value for the plot. Default is 10.

r_sqr	The lower limit of acceptable R^2 values (any curves below it will be flagged). Default is 0.5.
beads	The maximum multiplier for the curve fit. Any files with higher curve fit multipliers will be flagged as bead runs. If this argument is included, files with "runBeads" marked as TRUE in the header file will also be flagged as a bead run. Optional.
bubbles	The minimum difference between the starting ESD and the ESD with the most targets. Any files with a difference higher than this threshold will be flagged as mostly bubbles. Optional.
incomplete	A tuple with the minimum volume of cells (in c/L) and the minimum mL analyzed for a complete run. Any files with values below these thresholds will be flagged as incomplete. Optional.
missing_cells	The minimum image count to trigger count ratio. Any files with lower ratios will be flagged as missing cells. Optional.
biomass	The minimum number of targets in the most populated ESD bin for any given run. Any files with fewer targets will be flagged as having low biomass. Optional.
bloom	The minimum difference between the starting ESD and the ESD with the most targets. Any files with a difference less than this threshold will be flagged as a bloom. Will likely be lower than the bubbles threshold. Optional.
humidity	The maximum percent humidity. Any files with higher values will be flagged as high humidity. Optional.
micron_factor	The conversion factor to microns. Default is 1/3.4.

Details

The PSD function originates from the PSD python repository (Hayashi et al. in prep), which can be found at https://github.com/kudelalab/PSD. This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using ifcb_py_install. The function requires v2 features generated by the ifcb-analysis MATLAB package (Sosik and Olson 2007) found at https://github.com/hsosik/ifcb-analysis.

Value

A list with data, fits, and flags DataFrames if save_data is FALSE; otherwise, NULL.

References

Hayashi, K., Walton, J., Lie, A., Smith, J. and Kudela M. Using particle size distribution (PSD) to automate imaging flow cytobot (IFCB) data quality in coastal California, USA. In prep. Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

ifcb_py_install https://github.com/kudelalab/PSD https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Initialize the python session if not already set up
ifcb_py_install()

ifcb_psd(
  feature_folder = 'path/to/features',
  hdr_folder = 'path/to/hdr_data',
  save_data = TRUE,
  output_file = 'psd/svea_2021',
  plot_folder = 'psd/plots',
  use_marker = FALSE,
  start_fit = 13,
  r_sqr = 0.5,
  beads = 10 ** 9,
  bubbles = 150,
  incomplete = c(1500, 3),
  missing_cells = 0.7,
  biomass = 1000,
  bloom = 5,
  humidity = NULL,
  micron_factor = 1/3.0
)

## End(Not run)
```

---

ifcb_psd_plot *Generate PSD Plot for a Given Sample*

Description

This function generates a plot for a given sample from Particle Size Distribution (PSD) data and fits from Imaging FlowCytobot (IFCB). The PSD data and fits can be generated by `ifcb_psd` (Hayashi et al. in prep).

Usage

```
ifcb_psd_plot(sample_name, data, fits, start_fit)
```

Arguments

`sample_name`	The name of the sample to plot in DYYYYMMDDTHHMMSS.
`data`	A data frame containing the PSD data (data output from `ifcb_psd`), where each row represents a sample and each column represents different particle sizes in micrometers.
`fits`	A data frame containing the fit parameters for the power curve (fits output from `ifcb_psd`), where each row represents a sample and the columns include the parameters `a`, `k`, and `R2`.

start_fit The x-value threshold below which data should be excluded from the plot and fit.

Value

A ggplot object representing the PSD plot for the sample.

References

Hayashi, K., Walton, J., Lie, A., Smith, J. and Kudela M. Using particle size distribution (PSD) to automate imaging flow cytobot (IFCB) data quality in coastal California, USA. In prep.

See Also

ifcb_psd https://github.com/kudelalab/PSD

Examples

```
## Not run:
# Analyze PSD
psd <- ifcb_psd(feature_folder = 'path/to/features',
                hdr_folder = 'path/to/hdr_data',
                save_data = TRUE,
                output_file = 'psd/svea_2021',
                plot_folder = NULL,
                use_marker = FALSE,
                start_fit = 13,
                r_sqr = 0.5,
                beads = 10 ** 9,
                bubbles = 150,
                incomplete = c(1500, 3),
                missing_cells = 0.7,
                biomass = 1000,
                bloom = 5,
                humidity = NULL)

# Plot PSD of the first sample
plot <- ifcb_psd_plot(sample_name = "D20230316T101514",
                      data = psd$data,
                      fits = psd$fits,
                      start_fit = 10)
# Inspect plot
print(plot)

## End(Not run)
```

`ifcb_py_install` *Install iRfcb Python Environment*

Description

This function sets up the Python environment for `iRfcb`. By default, it creates and activates a Python virtual environment (`venv`) named "iRfcb" and installs the required Python packages from the "requirements.txt" file. Alternatively, users can opt to use the system Python instead of creating a virtual environment by setting `use_venv = FALSE`.

Usage

```
ifcb_py_install(
  envname = ".virtualenvs/iRfcb",
  use_venv = TRUE,
  packages = NULL
)
```

Arguments

`envname`	A character string specifying the name of the virtual environment to create. Default is ".virtualenvs/iRfcb".
`use_venv`	Logical. If `TRUE` (default), a virtual environment is created. If `FALSE`, the system Python is used instead, and missing packages are installed globally for the user.
`packages`	A character vector of additional Python packages to install. If NULL (default), only the packages from "requirements.txt" are installed.

Value

No return value. This function is called for its side effect of configuring the Python environment.

Examples

```
## Not run:
# Install the iRfcb Python environment using a virtual environment (default)
ifcb_py_install()

# Install the iRfcb Python environment with additional packages
ifcb_py_install(packages = c("numpy", "plotly"))

# Use system Python instead of a virtual environment
ifcb_py_install(use_venv = FALSE)

## End(Not run)
```

ifcb_read_features *Read Feature Files from a Specified Folder or File Paths*

Description

This function reads feature files from a given folder or a specified set of file paths, optionally filtering them based on whether they are multiblob or single blob files.

Usage

```
ifcb_read_features(feature_files = NULL, multiblob = FALSE, verbose = TRUE)
```

Arguments

feature_files	A path to a folder containing feature files or a character vector of file paths.
multiblob	Logical indicating whether to filter for multiblob files (default: FALSE).
verbose	Logical. Whether to display progress information. Default is TRUE.

Value

A named list of data frames, where each element corresponds to a feature file read from `feature_files`. The list is named with the base names of the feature files.

Examples

```
## Not run:
# Read feature files from a folder
features <- ifcb_read_features("path/to/feature_folder")

# Read only multiblob feature files
multiblob_features <- ifcb_read_features("path/to/feature_folder", multiblob = TRUE)

# Read feature files from a list of file paths
features <- ifcb_read_features(c("path/to/file1.csv", "path/to/file2.csv"))

## End(Not run)
```

ifcb_read_hdr_data *Reads HDR Data from IFCB HDR Files*

Description

This function reads all IFCB instrument settings information files (.hdr) from a specified directory.

Usage

```
ifcb_read_hdr_data(
  hdr_files,
  gps_only = FALSE,
  verbose = TRUE,
  hdr_folder = deprecated()
)
```

Arguments

hdr_files	A character string specifying the path to feature files or a folder path.
gps_only	A logical value indicating whether to include only GPS information (latitude and longitude). Default is FALSE.
verbose	A logical value indicating whether to print progress messages. Default is TRUE.
hdr_folder	**[Deprecated]** Use hdr_files instead.

Value

A data frame with sample names, GPS latitude, GPS longitude, and optionally timestamps.

Examples

```
## Not run:
# Extract all HDR data
hdr_data <- ifcb_read_hdr_data("path/to/data")
print(hdr_data)

# Extract only GPS data
gps_data <- ifcb_read_hdr_data("path/to/data", gps_only = TRUE)
print(gps_data)

## End(Not run)
```

---

ifcb_read_mat *Read a MATLAB .mat File in R*

Description

This function reads a MATLAB .mat file using a Python function via reticulate.

Usage

```
ifcb_read_mat(file_path)
```

Arguments

file_path	A character string representing the full path to the .mat file.

Details

This function requires a Python interpreter with SciPy installed.

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using ifcb_py_install.

Value

A list containing the MATLAB variables.

See Also

ifcb_py_install

Examples

```
## Not run:
data <- read_mat_file_r("C:/data/sample.mat")

## End(Not run)
```

---

ifcb_read_summary *Read and Summarize Classified IFCB Data*

Description

This function reads a MATLAB .mat file containing aggregated and classified IFCB (Imaging FlowCytobot) data generated by the countcells_allTBnew_user_training function from the ifcb-analysis repository (Sosik and Olson 2007), or a list of classified data generated by ifcb_summarize_class_counts. It returns a data frame with species counts and optionally biovolume information based on specified thresholds.

Usage

```
ifcb_read_summary(
  summary,
  hdr_directory = NULL,
  biovolume = FALSE,
  threshold = "opt",
  use_python = FALSE
)
```

Arguments

summary	A character string specifying the path to the .mat summary file or a list generated by ifcb_summarize_class_counts.
hdr_directory	A character string specifying the path to the directory containing header (.hdr) files. Default is NULL.
biovolume	A logical indicating whether the file contains biovolume data. Default is FALSE.
threshold	A character string specifying the threshold type for counts and biovolume. Options are "opt" (default), "adhoc", and "none".
use_python	Logical. If TRUE, attempts to read the .mat file using a Python-based method. Default is FALSE.

Details

If use_python = TRUE, the function tries to read the .mat file using ifcb_read_mat(), which relies on SciPy. This approach may be faster than the default approach using R.matlab::readMat(), especially for large .mat files. To enable this functionality, ensure Python is properly configured with the required dependencies. You can initialize the Python environment and install necessary packages using ifcb_py_install().

If use_python = FALSE or if SciPy is not available, the function falls back to using R.matlab::readMat().

Value

A data frame containing the summary information including file list, volume analyzed, species counts, optionally biovolume, and other metadata.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
summary_data <- ifcb_read_summary("path/to/summary_file.mat", biovolume = TRUE, threshold = "opt")
print(summary_data)

## End(Not run)
```

`ifcb_replace_mat_values` *Replace Values in MATLAB Classlist*

Description

This function replaces a target class ID with a new ID in MATLAB classlist files, generated by the code in the `ifcb-analysis` repository (Sosik and Olson 2007).

Usage

```
ifcb_replace_mat_values(
  manual_folder,
  out_folder,
  target_id,
  new_id,
  column_index = 1,
  do_compression = TRUE
)
```

Arguments

`manual_folder`	A character string specifying the path to the folder containing MAT classlist files to be updated.
`out_folder`	A character string specifying the path to the folder where updated MAT classlist files will be saved.
`target_id`	The target class ID to be replaced.
`new_id`	The new class ID to replace the target ID.
`column_index`	An integer value specifying which classlist column to edit. Default is 1 (manual).
`do_compression`	A logical value indicating whether to compress the .mat file. Default is TRUE.

Details

This function requires a python interpreter to be installed. The required python packages can be installed in a virtual environment using `ifcb_py_install`.

Value

This function does not return any value; it updates the classlist files in the specified directory.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

ifcb_py_install https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Initialize a python session if not already set up
ifcb_py_install()

# Replace class ID 99 with 1 in .mat classlist files
ifcb_replace_mat_values("output/manual", "output/manual", 99, 1, column_index = 1)

## End(Not run)
```

---

ifcb_run_image_gallery

Run IFCB Image Gallery

Description

Launches a Shiny application that provides an interactive interface for browsing and managing IFCB (Imaging FlowCytobot) image galleries.

Usage

```
ifcb_run_image_gallery()
```

Details

Users can specify a folder containing .png images, navigate through the images, select and unselect images, and download a list of selected images. This feature is particularly useful for quality control of annotated images. A downloaded list of images from the app can also be uploaded to filter and view only the selected images.

Value

No return value. This function launches a Shiny application for interactive image browsing and management.

Examples

```
## Not run:
# Run the IFCB image gallery Shiny app
ifcb_run_image_gallery()

## End(Not run)
```

`ifcb_summarize_biovolumes` *Summarize Biovolumes and Carbon Content from IFCB Data*

Description

This function calculates aggregated biovolumes and carbon content from Imaging FlowCytobot (IFCB) samples based on biovolume information from feature files. Images are grouped into classes either based on MATLAB classification, manually annotated files, or a user-supplied list of images and their corresponding class labels (e.g. from a CNN model).

Usage

```
ifcb_summarize_biovolumes(
  feature_folder,
  mat_folder = NULL,
  class2use_file = NULL,
  hdr_folder = NULL,
  custom_images = NULL,
  custom_classes = NULL,
  micron_factor = 1/3.4,
  diatom_class = "Bacillariophyceae",
  marine_only = FALSE,
  threshold = "opt",
  feature_recursive = TRUE,
  mat_recursive = TRUE,
  hdr_recursive = TRUE,
  use_python = FALSE,
  verbose = TRUE
)
```

Arguments

`feature_folder`	Path to the folder containing feature files (e.g., CSV format).
`mat_folder`	(Optional) Path to the folder containing MATLAB classification or manual annotation files.
`class2use_file`	(Optional) A character string specifying the path to the file containing the class2use variable (default NULL). Only needed when summarizing manual MATLAB results.
`hdr_folder`	(Optional) Path to the folder containing HDR files. Needed for calculating cell, biovolume and carbon concentration per liter.
`custom_images`	(Optional) A character vector of image filenames in the format DYYYYMMDDTHHMMSS_IFCBXXX_ZZZZZ, where "XXX" represents the IFCB number and "ZZZZZ" represents the ROI number. These filenames should match the `roi_number` assignment in the `feature_files` and can be used as a substitute for MATLAB files.

custom_classes	(Optional) A character vector of corresponding class labels for custom_images.
micron_factor	Conversion factor from microns per pixel (default: 1/3.4).
diatom_class	A string vector of diatom class names in the World Register of Marine Species (WoRMS). Default is "Bacillariophyceae".
marine_only	Logical. If TRUE, restricts the WoRMS search to marine taxa only. Default is FALSE.
threshold	Threshold for classification (default: "opt").
feature_recursive	Logical. If TRUE, the function will search for feature files recursively within the feature_folder. Default is TRUE.
mat_recursive	Logical. If TRUE, the function will search for MATLAB files recursively within the mat_folder. Default is TRUE.
hdr_recursive	Logical. If TRUE, the function will search for HDR files recursively within the hdr_folder (if provided). Default is TRUE.
use_python	Logical. If TRUE, attempts to read the .mat file using a Python-based method. Default is FALSE.
verbose	A logical indicating whether to print progress messages. Default is TRUE.

Details

This function performs the following steps:

1. Extracts biovolumes and carbon content from feature and classification results using ifcb_extract_biovolumes.
2. Optionally incorporates volume data from HDR files to calculate volume analyzed per sample.
3. Computes biovolume and carbon content per liter of sample analyzed.

The MATLAB classification or manual annotation files are generated by the ifcb-analysis repository (Sosik and Olson 2007). Users can optionally provide a **custom classification** by supplying a vector of image filenames (custom_images) along with corresponding class labels (custom_classes). This allows summarization of biovolume and carbon content without requiring MATLAB classification or manual annotation files (e.g. results from a CNN model).

Biovolumes are converted to carbon according to Menden-Deuer and Lessard 2000 for individual regions of interest (ROI), applying different conversion factors to diatoms and non-diatom protists. If provided, the function also incorporates sample volume data from HDR files to compute biovolume and carbon content per liter of sample.

If use_python = TRUE, the function tries to read the .mat file using ifcb_read_mat(), which relies on SciPy. This approach may be faster than the default approach using R.matlab::readMat(), especially for large .mat files. To enable this functionality, ensure Python is properly configured with the required dependencies. You can initialize the Python environment and install necessary packages using ifcb_py_install().

Value

A data frame summarizing aggregated biovolume and carbon content per class per sample. Columns include 'sample', 'classifier', 'class', 'biovolume_mm3', 'carbon_ug', 'ml_analyzed', 'biovolume_mm3_per_liter', and 'carbon_ug_per_liter'.

References

Menden-Deuer Susanne, Lessard Evelyn J., (2000), Carbon to volume relationships for dinoflagellates, diatoms, and other protist plankton, Limnology and Oceanography, 3, doi: 10.4319/lo.2000.45.3.0569.

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

Examples

```
## Not run:
# Example usage:
ifcb_summarize_biovolumes("path/to/features", "path/to/mat", hdr_folder = "path/to/hdr")

# Using custom classification result:
images <- c("D20220522T003051_IFCB134_00002",
            "D20220522T003051_IFCB134_00003")
classes = c("Mesodinium_rubrum",
            "Mesodinium_rubrum")

ifcb_summarize_biovolumes(feature_folder = "path/to/features",
                          hdr_folder = "path/to/hdr",
                          custom_images = images,
                          custom_classes = classes)

## End(Not run)
```

---

ifcb_summarize_class_counts

Count Cells from TreeBagger Classifier Output

Description

This function summarizes class results for a series of classifier output files and returns a summary data list.

Usage

```
ifcb_summarize_class_counts(
  classpath_generic,
  hdr_folder,
  year_range,
  use_python = FALSE
)
```

Arguments

classpath_generic	Character string specifying the location of the classifier output files. The path should include 'xxxx' in place of the 4-digit year (e.g., 'classxxxx_v1/').
hdr_folder	Character string specifying the directory where the data (hdr files) are located. This can be a URL for web services or a full path for local files.
year_range	Numeric vector specifying the range of years (e.g., 2013:2014) to process.
use_python	Logical. If TRUE, attempts to read the .mat file using a Python-based method. Default is FALSE.

Details

If use_python = TRUE, the function tries to read the .mat file using ifcb_read_mat(), which relies on SciPy. This approach may be faster than the default approach using R.matlab::readMat(), especially for large .mat files. To enable this functionality, ensure Python is properly configured with the required dependencies. You can initialize the Python environment and install necessary packages using ifcb_py_install().

If use_python = FALSE or if SciPy is not available, the function falls back to using R.matlab::readMat().

Value

A list containing the following elements:

class2useTB	Classes used in the TreeBagger classifier.
classcountTB	Counts of each class considering each target placed in the winning class.
classcountTB_above_optthresh	Counts of each class considering only classifications above the optimal threshold for maximum accuracy.
ml_analyzedTB	Volume analyzed for each file.
mdateTB	Dates associated with each file.
filelistTB	List of files processed.
classpath_generic	The generic classpath provided as input.
classcountTB_above_adhocthresh (optional)	Counts of each class considering only classifications above the adhoc threshold.
adhocthresh (optional)	The adhoc threshold used for classification.

Examples

```
## Not run:
ifcb_summarize_class_counts('C:/work/IFCB/user_training_test_data/class/classxxxx_v1/',
                            'C:/work/IFCB/user_training_test_data/data/', 2014)

## End(Not run)
```

`ifcb_summarize_png_counts` *Summarize Image Counts by Class and Sample*

Description

This function summarizes the number of images per class for each sample and timestamps, and optionally retrieves GPS positions, and IFCB information using `ifcb_read_hdr_data` and `ifcb_convert_filenames` functions.

Usage

```
ifcb_summarize_png_counts(
  png_folder,
  hdr_folder = NULL,
  sum_level = "sample",
  verbose = TRUE
)
```

Arguments

`png_folder`	A character string specifying the path to the main directory containing subfolders (classes) with .png images.
`hdr_folder`	A character string specifying the path to the directory containing the .hdr files. Default is NULL.
`sum_level`	A character string specifying the level of summarization. Options: "sample" (default) or "class".
`verbose`	A logical indicating whether to print progress messages. Default is TRUE.

Value

If sum_level is "sample", returns a data frame with columns: sample, ifcb_number, class_name, n_images, gpsLatitude, gpsLongitude, timestamp, year, month, day, time, roi_numbers. If sum_level is "class", returns a data frame with columns: class_name, n_images.

See Also

`ifcb_read_hdr_data` `ifcb_convert_filenames`

Examples

```
## Not run:
# Example usage:
# Assuming the following directory structure:
# path/to/png_folder/
# |- class1/
# |  |- sample1_00001.png
```

```
# |  |- sample1_00002.png
# |  |- sample2_00001.png
# |- class2/
# |  |- sample1_00003.png
# |  |- sample3_00001.png

png_folder <- "path/to/png_folder"
hdr_folder <- "path/to/hdr_folder" # This folder should contain corresponding .hdr files

# Summarize by sample
summary_sample <- ifcb_summarize_png_counts(png_folder,
                                            hdr_folder,
                                            sum_level = "sample",
                                            verbose = TRUE)
print(summary_sample)

# Summarize by class
summary_class <- ifcb_summarize_png_counts(png_folder,
                                           hdr_folder,
                                           sum_level = "class",
                                           verbose = TRUE)
print(summary_class)

## End(Not run)
```

`ifcb_summarize_png_metadata`
Summarize PNG Image Metadata

Description

This function processes IFCB data by reading images, matching them to the corresponding header and feature files, and joining them into a single dataframe. This function may be useful when preparing metadata files for an Ecotaxa submission.

Usage

```
ifcb_summarize_png_metadata(
  png_folder,
  feature_folder = NULL,
  hdr_folder = NULL
)
```

Arguments

`png_folder`	Character. The file path to the folder containing the PNG images.
`feature_folder`	Character. The file path to the folder containing the feature files (optional).
`hdr_folder`	Character. The file path to the folder containing the header files (optional).

Value

A dataframe that joins image data, header data, and feature data based on the sample and roi number.

Examples

```
## Not run:
png_folder <- "path/to/pngs"
feature_folder <- "path/to/features"
hdr_folder <- "path/to/hdr_data"
result_df <- ifcb_summarize_png_metadata(png_folder, feature_folder, hdr_folder)

## End(Not run)
```

---

ifcb_volume_analyzed *Estimate Volume Analyzed from IFCB Header File*

Description

This function reads an IFCB header file to extract sample run time and inhibittime, and returns the associated estimate of sample volume analyzed (in milliliters). The function assumes a standard IFCB configuration with a sample syringe operating at 0.25 mL per minute. For IFCB instruments after 007 and higher (except 008). This is the R equivalent function of `IFCB_volume_analyzed` from the `ifcb-analysis` repository (Sosik and Olson 2007).

Usage

```
ifcb_volume_analyzed(hdr_file, hdrOnly_flag = FALSE, flowrate = 0.25)
```

Arguments

`hdr_file`	A character vector specifying the path(s) to one or more .hdr files or URLs.
`hdrOnly_flag`	An optional flag indicating whether to skip ADC file estimation (default is FALSE).
`flowrate`	Milliliters per minute for syringe pump (default is 0.25).

Value

A numeric vector containing the estimated sample volume analyzed for each header file.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Example: Estimate volume analyzed from an IFCB header file
hdr_file <- "path/to/IFCB_hdr_file.hdr"
ml_analyzed <- ifcb_volume_analyzed(hdr_file)
print(ml_analyzed)

## End(Not run)
```

`ifcb_volume_analyzed_from_adc`
Estimate Volume Analyzed from IFCB ADC File

Description

This function reads an IFCB ADC file to extract sample run time and inhibittime, and returns the associated estimate of sample volume analyzed (in milliliters). The function assumes a standard IFCB configuration with a sample syringe operating at 0.25 mL per minute. For IFCB instruments after 007 and higher (except 008). This is the R equivalent function of `IFCB_volume_analyzed_fromADC` from the `ifcb-analysis repository` (Sosik and Olson 2007).

Usage

```
ifcb_volume_analyzed_from_adc(adc_file)
```

Arguments

`adc_file`	A character vector specifying the path(s) to one or more .adc files or URLs.

Value

A list containing:

- **ml_analyzed**: A numeric vector of estimated sample volume analyzed for each ADC file.
- **inhibittime**: A numeric vector of inhibittime values extracted from ADC files.
- **runtime**: A numeric vector of runtime values extracted from ADC files.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216.

See Also

https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
# Example: Estimate volume analyzed from an IFCB ADC file
adc_file <- "path/to/IFCB_adc_file.csv"
adc_info <- ifcb_volume_analyzed_from_adc(adc_file)
print(adc_info$ml_analyzed)

## End(Not run)
```

`ifcb_which_basin` *Determine if Points are in a Specified Sea Basin*

Description

This function identifies which sub-basin a set of latitude and longitude points belong to, using a user-specified or default shapefile. The default shapefile includes the Baltic Sea, Kattegat, and Skagerrak basins and is included in the `iRfcb` package.

Usage

```
ifcb_which_basin(latitudes, longitudes, plot = FALSE, shape_file = NULL)
```

Arguments

`latitudes`	A numeric vector of latitude points.
`longitudes`	A numeric vector of longitude points.
`plot`	A boolean indicating whether to plot the points along with the sea basins. Default is FALSE.
`shape_file`	The absolute path to a custom polygon shapefile in WGS84 (EPSG:4326) that represents the sea basin. Defaults to the Baltic Sea, Kattegat, and Skagerrak basins included in the `iRfcb` package.

Details

This function reads a pre-packaged shapefile of the Baltic Sea, Kattegat, and Skagerrak basins from the `iRfcb` package by default, or a user-supplied shapefile if provided. The shapefiles originate from SHARK (https://shark.smhi.se/). It sets the CRS, transforms the CRS to WGS84 (EPSG:4326) if necessary, and checks if the given points fall within the specified sea basin. Optionally, it plots the points and the sea basin polygons together.

Value

A vector indicating the basin each point belongs to, or a ggplot object if `plot = TRUE`.

Examples

```
# Define example latitude and longitude vectors
latitudes <- c(55.337, 54.729, 56.311, 57.975)
longitudes <- c(12.674, 14.643, 12.237, 10.637)

# Check in which Baltic sea basin the points are in
points_in_the_baltic <- ifcb_which_basin(latitudes, longitudes)
print(points_in_the_baltic)

# Plot the points and the basins
ifcb_which_basin(latitudes, longitudes, plot = TRUE)
```

---

ifcb_zip_matlab *Create a Zip Archive of Manual MATLAB Files*

Description

This function creates a zip archive containing specified files and directories for manually annotated IFCB images, organized into a structured format suitable for distribution or storage. The MATLAB files are generated by the `ifcb-analysis` repository (Sosik and Olson 2007). The zip archive can be used to submit IFCB data to repositories like in the SMHI IFCB Plankton Image Reference Library (Torstensson et al., 2024).

Usage

```
ifcb_zip_matlab(
  manual_folder,
  features_folder,
  class2use_file,
  zip_filename,
  data_folder = NULL,
  readme_file = NULL,
  matlab_readme_file = NULL,
  email_address = "",
  version = "",
  print_progress = TRUE,
  feature_recursive = TRUE,
  manual_recursive = FALSE,
  data_recursive = TRUE,
  quiet = FALSE
)
```

Arguments

`manual_folder` The directory containing .mat files to be included in the zip archive.

`features_folder`
: The directory containing .csv files, including subfolders, to be included in the zip archive.

`class2use_file`
: The path to the file (class2use_file) that will be renamed and included in the 'config' directory of the zip archive.

`zip_filename`
: The filename for the zip archive to be created.

`data_folder`
: Optionally, the directory containing additional data files (.roi, .adc, .hdr) to be included in the zip archive.

`readme_file`
: Optionally, the path to a README file that will be updated with metadata and included in the zip archive.

`matlab_readme_file`
: Optionally, the path to a MATLAB README file whose content will be appended to the end of the README file in the zip archive.

`email_address`
: The email address to be included in the README file for contact information.

`version`
: Optionally, the version number to be included in the README file.

`print_progress`
: A logical value indicating whether to print progress bar. Default is TRUE.

`feature_recursive`
: Logical. If TRUE, the function will search for feature files recursively within the `feature_folder`. Default is TRUE.

`manual_recursive`
: Logical. If TRUE, the function will search for MATLAB files recursively within the `manual_folder`. Default is FALSE.

`data_recursive`
: Logical. If TRUE, the function will search for data files recursively within the `data_folder` (if provided). Default is TRUE.

`quiet`
: Logical. If TRUE, suppresses messages about the progress and completion of the zip process. Default is FALSE.

Details

This function performs the following operations:

- Lists `.mat` files from `manual_folder`.
- Lists `.csv` files from `features_folder` (including subfolders).
- Lists `.roi`, `.adc`, `.hdr` files from `data_folder` if provided.
- Copies listed files to temporary directories (`manual_dir`, `features_dir`, `data_dir`, `config_dir`).
- Renames and copies `class2use_file` to `config_dir` as `class2use.mat`.
- Updates `readme_file` with metadata (if provided) and appends PNG image statistics and MATLAB README content.
- Creates a manifest file (`MANIFEST.txt`) listing all files in the zip archive.
- Creates a zip archive (`zip_filename`) containing all copied and updated files.
- Cleans up temporary directories after creating the zip archive.

Value

No return value. This function creates a zip archive containing the specified files and directories.

References

Sosik, H. M. and Olson, R. J. (2007), Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr: Methods 5, 204–216. Torstensson, Anders; Skjevik, Ann-Turi; Mohlin, Malin; Karlberg, Maria; Karlson, Bengt (2024). SMHI IFCB Plankton Image Reference Library. SciLifeLab. Dataset. doi:10.17044/scilifelab.25883455

See Also

ifcb_zip_pngs https://github.com/hsosik/ifcb-analysis

Examples

```
## Not run:
ifcb_zip_matlab("path/to/manual_files", "path/to/feature_files",
                "path/to/class2use.mat", "output_zip_archive.zip",
                data_folder = "path/to/data_files",
              readme_file = system.file("exdata/README-template.md", package = "iRfcb"),
                matlab_readme_file = system.file("inst/exdata/MATLAB-template.md",
                                                 package = "iRfcb"),
                email_address = "example@email.com",
                version = "1.0")

## End(Not run)
```

---

ifcb_zip_pngs *Zip PNG Folders*

Description

This function zips directories containing .png files and optionally includes README and MANIFEST files. It can also split the resulting zip file into smaller parts if it exceeds a specified size. The zip archive can be used to submit IFCB data to repositories like in the SMHI IFCB Plankton Image Reference Library (Torstensson et al., 2024).

Usage

```
ifcb_zip_pngs(
  png_folder,
  zip_filename,
  readme_file = NULL,
  email_address = "",
  version = "",
  print_progress = TRUE,
  include_txt = FALSE,
  split_zip = FALSE,
  max_size = 500,
  quiet = FALSE
)
```

Arguments

png_folder	The directory containing subdirectories with .png files.
zip_filename	The name of the zip file to create.
readme_file	Optional path to a README file for inclusion in the zip package.
email_address	Optional email address to include in the README file.
version	Optional version information to include in the README file.
print_progress	A logical value indicating whether to print progress bar. Default is TRUE.
include_txt	A logical value indicating whether to include text (.txt, .tsv and .csv) files located in the subdirectories. Default is FALSE.
split_zip	A logical value indicating whether to split the zip file into smaller parts if its size exceeds max_size. Default is FALSE.
max_size	The maximum size (in MB) for the zip file before it gets split. Only used if split_zip is TRUE. Default is 500 MB.
quiet	Logical. If TRUE, suppresses messages about the progress and completion of the zip process. Default is FALSE.

Value

This function does not return any value; it creates a zip archive and optionally splits it into smaller files if specified.

References

Torstensson, Anders; Skjevik, Ann-Turi; Mohlin, Malin; Karlberg, Maria; Karlson, Bengt (2024). SMHI IFCB Plankton Image Reference Library. SciLifeLab. Dataset. doi:10.17044/scilifelab.25883455

See Also

ifcb_zip_matlab

Examples

```
## Not run:
# Zip all subdirectories in the 'images' folder with a README file
ifcb_zip_pngs("path/to/images",
              "images.zip",
              readme_file = system.file("exdata/README-template.md", package = "iRfcb"),
              email_address = "example@example.com",
              version = "1.0")

# Zip all subdirectories in the 'images' folder without a README file
ifcb_zip_pngs("path/to/images", "images.zip")

## End(Not run)
```

read_hdr_file *Function to Read Individual Files and Extract Relevant Lines*

Description

This function reads an HDR file and extracts relevant lines containing parameters and their values.

Usage

```
read_hdr_file(file)
```

Arguments

file	A character string specifying the path to the HDR file.

Value

A data frame with columns: parameter, value, and file.

split_large_zip *Split Large Zip File into Smaller Parts*

Description

This helper function takes an existing zip file, extracts its contents, and splits it into smaller zip files without splitting subfolders.

Usage

```
split_large_zip(zip_file, max_size = 500, quiet = FALSE)
```

Arguments

zip_file	The path to the large zip file.
max_size	The maximum size (in MB) for each split zip file. Default is 500 MB.
quiet	Logical. If TRUE, suppresses messages about the progress and completion of the zip process. Default is FALSE.

Value

This function does not return any value; it creates multiple smaller zip files.

Examples

```
## Not run:
# Split an existing zip file into parts of up to 500 MB
split_large_zip("large_file.zip", max_size = 500)

## End(Not run)
```

---

summarize_TBclass *Summarize TreeBagger Classifier Results*

Description

This function reads a TreeBagger classifier result file (.mat format) and summarizes the number of targets in each class based on the classification scores and thresholds.

Usage

```
summarize_TBclass(classfile, adhocthresh = NULL)
```

Arguments

classfile	Character string specifying the path to the TreeBagger classifier result file (.mat format).
adhocthresh	Numeric vector specifying the adhoc thresholds for each class. If NULL (default), no adhoc thresholding is applied. If a single numeric value is provided, it is applied to all classes.

Value

A list containing three elements:

classcount	Numeric vector of counts for each class based on the winning class assignment.
classcount_above_optthresh	Numeric vector of counts for each class above the optimal threshold for maximum accuracy.
classcount_above_adhocthresh	Numeric vector of counts for each class above the specified adhoc thresholds (if provided).

vol2C_lgdiatom *Convert Biovolume to Carbon for Large Diatoms*

Description

This function converts biovolume in microns^3 to carbon in picograms for large diatoms (> 2000 micron^3) according to Menden-Deuer and Lessard 2000. The formula used is: log pgC cell^-1 = log a + b * log V (um^3), with log a = -0.933 and b = 0.881 for diatoms > 3000 um^3.

Usage

```
vol2C_lgdiatom(volume)
```

Arguments

volume	A numeric vector of biovolume measurements in microns^3.

Value

A numeric vector of carbon measurements in picograms.

Examples

```
# Volumes in microns^3
volume <- c(5000, 10000, 20000)

# Convert biovolume to carbon for large diatoms
vol2C_lgdiatom(volume)
```

vol2C_nondiatom *Convert Biovolume to Carbon for Non-Diatom Protists*

Description

This function converts biovolume in microns^3 to carbon in picograms for protists besides large diatoms (> 3000 micron^3) according to Menden-Deuer and Lessard 2000. The formula used is: log pgC cell^-1 = log a + b * log V (um^3), with log a = -0.665 and b = 0.939.

Usage

```
vol2C_nondiatom(volume)
```

Arguments

volume	A numeric vector of biovolume measurements in microns^3.

Value

A numeric vector of carbon measurements in picograms.

Examples

```
# Volumes in microns^3
volume <- c(5000, 10000, 20000)

# Convert biovolume to carbon for non-diatom protists
vol2C_nondiatom(volume)
```

Index