

Package ‘iRfcb’

March 1, 2026

Type Package

Title Tools for Managing Imaging FlowCytobot (IFCB) Data

Version 0.8.1

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/en/>, 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](https://doi.org/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

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

Depends R (>= 4.1)

Imports curl (>= 6.0.0), dplyr, ggplot2, lifecycle, lubridate, png, R.matlab, readr, reticulate (>= 1.41.0), sf, shiny, stringr, tidyr, worrms, zip, jsonlite

Suggests hdf5r, knitr, mockery, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

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

RoxygenNote 7.3.3

Config/Needs/website rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Anders Torstensson [aut, cre] (Swedish Meteorological and Hydrological Institute, ORCID: <<https://orcid.org/0000-0002-8283-656X>>),
 Kendra Hayashi [ctb] (ORCID: <<https://orcid.org/0000-0003-1600-9504>>),
 Jamie Enslein [ctb],
 Raphael Kudela [ctb] (ORCID: <<https://orcid.org/0000-0002-8640-1205>>),
 Alle Lie [ctb] (ORCID: <<https://orcid.org/0009-0001-8709-4841>>),
 Jayme Smith [ctb] (ORCID: <<https://orcid.org/0000-0002-9669-4427>>),
 DTO-BioFlow [fnd] (Horizon Europe, HORIZON-MISS-2022-OCEAN-01-07),
 SBDI [fnd] (Swedish Research Council, 2019-00242)

Maintainer Anders Torstensson <anders.torstensson@smhi.se>

Repository CRAN

Date/Publication 2026-03-01 13:30:08 UTC

Contents

create_package_manifest	3
ifcb_adjust_classes	4
ifcb_annotate_batch	5
ifcb_annotate_samples	7
ifcb_classify_images	9
ifcb_classify_models	11
ifcb_classify_sample	12
ifcb_convert_filenames	14
ifcb_correct_annotation	15
ifcb_count_mat_annotations	16
ifcb_create_class2use	18
ifcb_create_manifest	19
ifcb_create_manual_file	20
ifcb_download_dashboard_data	21
ifcb_download_dashboard_metadata	24
ifcb_download_test_data	25
ifcb_download_who_i_plankton	26
ifcb_extract_annotated_images	27
ifcb_extract_biovolumes	30
ifcb_extract_classified_images	33
ifcb_extract_pngs	35
ifcb_get_ecotaxa_example	37
ifcb_get_ferrybox_data	38
ifcb_get_mat_names	39
ifcb_get_mat_variable	40
ifcb_get_runtime	42
ifcb_get_shark_colnames	43

ifcb_get_shark_example	44
ifcb_get_trophic_type	44
ifcb_is_diatom	45
ifcb_is_in_basin	46
ifcb_is_near_land	47
ifcb_list_dashboard_bins	49
ifcb_match_taxa_names	50
ifcb_merge_manual	51
ifcb_prepare_who_i_plankton	53
ifcb_psd	56
ifcb_psd_plot	59
ifcb_py_install	61
ifcb_read_features	62
ifcb_read_hdr_data	63
ifcb_read_mat	64
ifcb_read_summary	65
ifcb_replace_mat_values	66
ifcb_run_image_gallery	68
ifcb_save_classification	69
ifcb_summarize_biovolumes	71
ifcb_summarize_class_counts	74
ifcb_summarize_png_counts	75
ifcb_summarize_png_metadata	77
ifcb_volume_analyzed	78
ifcb_volume_analyzed_from_adc	79
ifcb_which_basin	80
ifcb_zip_images_by_class	81
ifcb_zip_matlab	82
ifcb_zip_pngs	84
process_ifcb_string	86
read_hdr_file	86
split_large_zip	87
summarize_TBclass	88
vol2C_lgdiatom	88
vol2C_nondiato_m	89
Index	90

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

Python must be installed to use this function. 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:  
# Initialize a python session if not already set up  
ifcb_py_install()  
  
ifcb_adjust_classes("data/config/class2use.mat",  
                  "data/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 files 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_files,  
  class2use_file,  
  manual_output = NULL,  
  manual_recursive = FALSE,  
  unclassified_id = 1,  
  do_compression = TRUE,  
  adc_folder = deprecated()  
)
```

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_files	A character string specifying the path to the folder containing the raw data, organized in subfolders by year (YYYY) and date (DYYYYMMDD), or a vector with full paths to the .adc files. Each ADC file is 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.
adc_folder	[Deprecated] Use adc_files instead.

Details

Python must be installed to use this function. 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_files.

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_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_files = "path/to/adc",
    class2use_file = "path/to/class2use.mat"
)

## End(Not run)
```

`ifcb_annotate_samples` *Create Manual Classification MAT Files from PNG Subfolders*

Description

This function creates manual classification `.mat` files compatible with the code in the `ifcb-analysis` MATLAB repository (Sosik and Olson 2007) by mapping ROIs to class IDs based on user-provided PNG images (organized into subfolders named after classes) and a `class2use` MAT file.

Usage

```
ifcb_annotate_samples(
    png_folder,
    adc_folder,
    class2use_file,
    output_folder,
```

```

    sample_names = NULL,
    unclassified_id = 1,
    remove_trailing_numbers = TRUE,
    do_compression = TRUE
)

```

Arguments

png_folder	Directory containing PNG images organized into subfolders named after classes. Each PNG file represents a single ROI extracted from an IFCB sample and must follow the standard IFCB naming convention (for example, "D20220712T210855_IFCB134_00042.png") which is used to map the image to the corresponding ROI index in the ADC file.
adc_folder	Directory containing ADC files for the samples.
class2use_file	Path to a class2use MAT file. This file should contain the vector of classes used for matching PNG annotations to class IDs.
output_folder	Directory where the resulting MAT files will be written. If the folder does not exist, it will be created automatically.
sample_names	Optional character vector of IFCB sample names (e.g., "D20220712T210855_IFCB134"). If NULL (default), all samples detected from the PNG filenames in png_folder will be processed. Each sample must have a corresponding ADC file in adc_folder.
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.
remove_trailing_numbers	Logical. If TRUE (default), trailing numeric suffixes are removed from PNG subfolder names before matching them to entries in class2use (for example, "Skeletonema_036" becomes "Skeletonema"). This is useful when class folders include numeric identifiers that are not part of the class names in class2use.
do_compression	A logical value indicating whether to compress the .mat file. Default is TRUE.

Details

Python must be installed to use this function. The required python packages can be installed in a virtual environment using `ifcb_py_install()`.

Each sample should have ADC files in `adc_folder` and corresponding PNG images stored in subfolders under `png_folder`, where each subfolder is named after a class (e.g., `Skeletonema`, `Dinophysis_acuminata`, `unclassified`). The function automatically maps PNG filenames to ROI indices, assigns class IDs based on `class2use`, and writes the resulting MAT file in `output_folder`.

- The function reads all PNG images in subfolders of `png_folder`, extracts class names from folder names, and converts PNG filenames to ROI indices using `ifcb_convert_filenames()`.
- Class IDs are assigned using `match()` against `class2use`. If any classes cannot be matched, a warning lists the unmatched classes and shows the `ifcb_get_mat_variable()` command to inspect available classes.
- The function writes one MAT file per sample using `ifcb_create_manual_file()`.

Value

Invisibly returns TRUE on successful completion.

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:
# Example: Annotate a single IFCB sample
sample_names <- "D20220712T210855_IFCB134"
png_folder <- "data/annotated_png_images/"
adc_folder <- "data/raw"
class2use_file <- "data/manual/class2use.mat"
output_folder <- "data/manual/"

# Create manual MAT file for this sample
ifcb_annotate_samples(
  png_folder = png_folder,
  adc_folder = adc_folder,
  class2use_file = class2use_file,
  output_folder = output_folder,
  sample_names = sample_names
)

## End(Not run)
```

`ifcb_classify_images` *Classify Pre-Extracted IFCB PNG Images Using a Gradio Application*

Description

Classifies one or more pre-extracted IFCB PNG images through a CNN model served by a Gradio application. Each PNG is uploaded to the Gradio server and the prediction result is returned as a data frame. Per-class F2 optimal thresholds are applied automatically; predictions scoring below the threshold for their class are labeled "unclassified" in `class_name`.

Usage

```
ifcb_classify_images(
  png_file,
  gradio_url = "https://irfcb-classify.hf.space",
  top_n = 1,
  model_name = "SMHI NIVA ResNet50 V5",
  verbose = TRUE
)
```

Arguments

png_file	A character vector of paths to PNG files to classify.
gradio_url	A character string specifying the base URL of the Gradio application. Default is "https://irfcb-classify.hf.space", which is an example Hugging Face Space with limited resources intended for testing and demonstration. For large-scale classification, deploy your own instance of the classification app (source code: https://github.com/EuropeanIFCBGroup/ifcb-inference-app) and pass its URL here.
top_n	An integer specifying the number of top predictions to return per image. Default is 1 (top prediction only). Use Inf to return all predictions.
model_name	A character string specifying the name of the CNN model to use for classification. Default is "SMHI NIVA ResNet50 V5". Use <code>ifcb_classify_models()</code> to list all available models.
verbose	A logical value indicating whether to print progress messages. Default is TRUE.

Details

To classify all images in a raw IFCB sample (.roi file) without first extracting them manually, use `ifcb_classify_sample()` instead.

Value

A data frame with the following columns:

file_name	The PNG file name of the classified image.
class_name	The predicted class name with per-class thresholds applied; "unclassified" if the score is below the threshold.
class_name_auto	The winning class name without any threshold applied (argmax of scores).
score	The prediction confidence score (0–1).
model_name	The name of the CNN model used for classification.

Images that could not be classified have NA in `class_name`, `class_name_auto`, and `score`. When `top_n > 1`, multiple rows are returned per image (one per prediction).

See Also

`ifcb_classify_sample()` to classify all images in a raw IFCB sample without prior extraction. `ifcb_classify_models()` to list available CNN models. `ifcb_extract_pngs()` to extract PNG images from IFCB ROI files.

Examples

```
## Not run:
# Classify a single pre-extracted PNG
result <- ifcb_classify_images("path/to/D20220522T003051_IFCB134_00001.png")

# Classify several PNGs at once
pngs <- list.files("path/to/png_folder", pattern = "\\*.png$",
                  full.names = TRUE)
result <- ifcb_classify_images(pngs, top_n = 3)

## End(Not run)
```

ifcb_classify_models *List Available CNN Models from a Gradio Classification Server*

Description

Queries the Gradio API to retrieve the names of all CNN models available for IFCB image classification. These model names can be passed to the `model_name` argument of `ifcb_classify_images()` and `ifcb_classify_sample()`.

Usage

```
ifcb_classify_models(gradio_url = "https://irfcb-classify.hf.space")
```

Arguments

`gradio_url` A character string specifying the base URL of the Gradio application. Default is "https://irfcb-classify.hf.space", which is an example Hugging Face Space with limited resources intended for testing and demonstration. For large-scale classification, deploy your own instance of the classification app with your own model (source code: <https://github.com/EuropeanIFCBGroup/ifcb-inference-app>) and pass its URL here.

Value

A character vector of available model names.

See Also

`ifcb_classify_images()`, `ifcb_classify_sample()`

Examples

```
## Not run:
# List available models
models <- ifcb_classify_models()
print(models)

# Use a specific model for classification
result <- ifcb_classify_images("image.png", model_name = models[1])

## End(Not run)
```

ifcb_classify_sample *Classify All Images in an IFCB Sample Using a Gradio Application*

Description

Extracts PNG images from an IFCB sample (.roi file) using `ifcb_extract_pngs()` into a temporary directory, then classifies each image through a CNN model served by a Gradio application. Per-class F2 optimal thresholds are applied automatically. The temporary directory is automatically removed when the function exits.

Usage

```
ifcb_classify_sample(
  roi_file,
  gradio_url = "https://irfcb-classify.hf.space",
  top_n = 1,
  model_name = "SMHI NIVA ResNet50 V5",
  verbose = TRUE,
  ...
)
```

Arguments

<code>roi_file</code>	A character string specifying the path to the .roi file.
<code>gradio_url</code>	A character string specifying the base URL of the Gradio application. Default is "https://irfcb-classify.hf.space", which is an example Hugging Face Space with limited resources intended for testing and demonstration. For large-scale classification, deploy your own instance of the classification app (source code: https://github.com/EuropeanIFCBGroup/ifcb-inference-app) and pass its URL here.
<code>top_n</code>	An integer specifying the number of top predictions to return per image. Default is 1 (top prediction only). Use Inf to return all predictions.
<code>model_name</code>	A character string specifying the name of the CNN model to use for classification. Default is "SMHI NIVA ResNet50 V5". Use <code>ifcb_classify_models()</code> to list all available models.

`verbose` A logical value indicating whether to print progress messages. Default is TRUE.
`...` Additional arguments passed to `ifcb_extract_pngs()` (e.g. `ROInumbers`, `gamma`, `scale_bar_um`).

Details

To classify individual pre-extracted PNG files, use `ifcb_classify_images()` directly.

Value

A data frame with the following columns:

`file_name` The PNG file name of the classified image.
`class_name` The predicted class name with per-class thresholds applied; "unclassified" if the score is below the threshold.
`class_name_auto` The winning class name without any threshold applied (argmax of scores).
`score` The prediction confidence score (0–1).
`model_name` The name of the CNN model used for classification.

Images that could not be classified have NA in `class_name`, `class_name_auto`, and `score`. When `top_n > 1`, multiple rows are returned per image (one per prediction).

See Also

`ifcb_classify_images()` to classify pre-extracted PNG files directly. `ifcb_classify_models()` to list available CNN models. `ifcb_extract_pngs()` to extract PNG images from IFCB ROI files.

Examples

```
## Not run:
# Classify all ROIs in a sample (top prediction per image)
result <- ifcb_classify_sample("path/to/D20220522T003051_IFCB134.roi")
head(result)

# Return top 3 predictions per image
result <- ifcb_classify_sample(
  "path/to/D20220522T003051_IFCB134.roi",
  top_n = 3
)

# Classify only specific ROI numbers
result <- ifcb_classify_sample(
  "path/to/D20220522T003051_IFCB134.roi",
  ROInumbers = c(1, 5, 10)
)

## 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(filenamees, tz = "UTC")
```

Arguments

<code>filenamees</code>	A character vector of IFCB filenames in the format "DYYYYMMDDTHH-MMSS_IFCBxxx" or "IFCBxxx_YYYY_DDD_HHMMSS". Filenames can optionally include an ROI number, which will be extracted if present.
<code>tz</code>	Character. Time zone to assign to the extracted timestamps. Defaults to "UTC". Set this to a different time zone if needed.

Value

A tibble with the following columns:

- `sample`: The extracted sample name (character).
- `full_timestamp`: The full timestamp in "YYYY-MM-DD HH:MM:SS" format (POSIXct).
- `year`: The year extracted from the timestamp (integer).
- `month`: The month extracted from the timestamp (integer).
- `day`: The day extracted from the timestamp (integer).
- `time`: The extracted time in "HH:MM:SS" format (character).
- `ifcb_number`: The IFCB instrument number (character).
- `roi`: The extracted ROI number if available (integer or NA).

If the `roi` column is empty (all NA), it will be excluded from the output.

Examples

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

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

Details

Python must be installed to use this function. 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.

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

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

Python must be installed to use this function. 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:
# Initialize a python session if not already set up
ifcb_py_install()

# Example usage:
classes <- c("unclassified", "Dinobryon_spp", "Helicostomella_spp")

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

## 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_create_manual_file
```

Create a Manual Classification MAT File

Description

Generates a .mat file for IFCB data with classification structure using a specified number of ROIs and class names. The output_file generated by this function is compatible with the code in the ifcb-analysis repository (Sosik and Olson 2007).

Usage

```
ifcb_create_manual_file(
  roi_length,
  class2use,
  output_file,
  classlist = 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.
classlist	Integer or numeric vector. Defines the values for the second column of the class list, typically representing the manual classification labels: <ul style="list-style-type: none"> • If a single value is provided, all rows will be assigned this value. For example, all ROIs can be assigned to class index 1 (default), which typically represents the unclassified category.

- If a numeric vector of the same length as `roi_length` is provided, the corresponding values will be used per row.

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

Details

Python must be installed to use this function. 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.

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:
# 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_manual_file(roi_length = 100,
                       class2use = c("unclassified", "Aphanizomenon_spp"),
                       output_file = "output.mat")

# Create a MAT file with 50 unclassified ROIs (1) and 50 Aphanizomenon_spp (2) ROIs
ifcb_create_manual_file(roi_length = 100,
                       class2use = c("unclassified", "Aphanizomenon_spp"),
                       output_file = "output.mat",
                       classlist = c(rep(1, 50), rep(2, 50)))

## End(Not run)
```

`ifcb_download_dashboard_data`

Download IFCB data files from an IFCB Dashboard

Description

This function downloads specified IFCB data files from a given IFCB Dashboard URL. It supports optional filename conversion and ADC file adjustments from the old IFCB file format.

Usage

```

ifcb_download_dashboard_data(
  dashboard_url,
  samples,
  file_types,
  dest_dir,
  convert_filenames = FALSE,
  convert_adc = FALSE,
  parallel_downloads = 5,
  sleep_time = 2,
  multi_timeout = 120,
  max_retries = 3,
  quiet = FALSE
)

```

Arguments

dashboard_url	Character. The base URL of the IFCB dashboard (e.g., "https://ifcb-data.who.i.edu"). If no subpath (e.g., /data/ or /mvco/) is included, /data/ will be added automatically. For the "features" and "autoclass" file_types, the dataset name needs to be included in the url (e.g. "https://ifcb-data.who.i.edu/mvco/").
samples	Character vector. The IFCB sample identifiers (e.g., "IFCB1_2014_188_222013" or "D20220807T025424_IFCB010").
file_types	Character vector. Specifies which file types to download. Allowed values: "blobs", "features", "autoclass", "roi", "zip", "hdr", "adc".
dest_dir	Character. The directory where downloaded files will be saved.
convert_filenames	Logical. If TRUE, converts filenames of the old format "IFCBxxx_YYYY_DDD_HHMMSS" to the new format (YYYYMMDDTHHMMSS_IFCBXXX or IYYYYMMDDTHHMMSS_IFCBXXX). Default is FALSE. [Experimental]
convert_adc	Logical. If TRUE, adjusts .adc files from older IFCB instruments (IFCB1–6, with filenames in the format "IFCBxxx_YYYY_DDD_HHMMSS") by inserting four empty columns after column 7 to match the newer format. Default is FALSE. [Experimental]
parallel_downloads	Integer. The number of files to download in parallel per batch. This helps manage network load and system performance. Default is 5.
sleep_time	A numeric value indicating the number of seconds to wait between each batch of downloads. Default is 2.
multi_timeout	Numeric. The maximum time in seconds that the curl multi-download request will wait for a response before timing out. This helps prevent hanging downloads in case of slow or unresponsive servers. Default is 120 seconds.
max_retries	An integer specifying the maximum number of attempts to retrieve data in case the server is unable to handle the request. Default is 3.
quiet	Logical. If TRUE, suppresses messages about the progress and completion of the download process. Default is FALSE.

Details

This function can download several files in parallel if the server allows it. The download parameters can be adjusted using the `parallel_downloads`, `sleep_time` and `multi_timeout` arguments.

If `convert_filenames = TRUE` [**Experimental**], filenames in the "IFCBxxx_YYYY_DDD_HHMMSS" format (used by IFCB1-6) will be converted to IYYYYMMDDTHHMMSS_IFCBXXX, ensuring compatibility with blob extraction in `ifcb-analysis` (Sosik & Olson, 2007), which identified the old .adc format by the first letter of the filename.

If `convert_adc = TRUE` [**Experimental**] and `convert_filenames = TRUE` [**Experimental**], the "IFCBxxx_YYYY_DDD_HHMMSS" format will instead be converted to DYYYYMMDDTHHMMSS_IFCBXXX. Additionally, .adc files will be modified to include four empty columns (PMT-A peak, PMT-B peak, PMT-C peak, and PMT-D peak), aligning them with the structure of modern .adc files for full compatibility with `ifcb-analysis`.

Value

This function does not return a value. It performs the following actions:

- Downloads the requested files into `dest_dir`.
- If `convert_adc = TRUE`, modifies ADC files in place by inserting four empty columns after column 7.
- Displays messages indicating the download status.

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_download_dashboard_metadata\(\)](#) to retrieve metadata from the IFCB Dashboard API.

[ifcb_list_dashboard_bins\(\)](#) to retrieve list of available bins from the IFCB Dashboard API.

Examples

```
ifcb_download_dashboard_data(  
  dashboard_url = "https://ifcb-data.whoi.edu/mvco/",  
  samples = "IFCB1_2014_188_222013",  
  file_types = c("blobs", "autoclass"),  
  dest_dir = tempdir(),  
  convert_filenames = FALSE,  
  convert_adc = FALSE,  
  quiet = TRUE  
)
```

`ifcb_download_dashboard_metadata`*Download metadata from the IFCB Dashboard API*

Description

Download metadata from the IFCB Dashboard API

Usage

```
ifcb_download_dashboard_metadata(base_url, dataset_name = NULL, quiet = FALSE)
```

Arguments

<code>base_url</code>	Character. Base URL to the IFCB Dashboard (e.g. "https://ifcb-data.who.edu/").
<code>dataset_name</code>	Optional character. Dataset slug (e.g. "mvco") to retrieve metadata for a specific dataset. If NULL, all available metadata are downloaded.
<code>quiet</code>	Logical. If TRUE, suppresses progress messages. Default is FALSE.

Value

A data frame containing the exported metadata.

See Also

[ifcb_download_dashboard_data\(\)](#) to download data from the IFCB Dashboard API.

[ifcb_list_dashboard_bins\(\)](#) to retrieve list of available bins from the IFCB Dashboard API.

Examples

```
# Download metadata for a specific dataset
metadata_mvco <- ifcb_download_dashboard_metadata("https://ifcb-data.who.edu/",
                                                dataset_name = "mvco",
                                                quiet = TRUE)

# Print result as tibble
print(metadata_mvco)
```

`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("introduction", package = "iRfcb")`.

Usage

```
ifcb_download_test_data(  
  dest_dir,  
  figshare_article = "48158716",  
  max_retries = 3,  
  sleep_time = 10,  
  keep_zip = FALSE,  
  verbose = TRUE,  
  expected_checksum = deprecated()  
)
```

Arguments

<code>dest_dir</code>	The destination directory where the files will be unzipped.
<code>figshare_article</code>	The file article number at the SciLifeLab Figshare data repository. By default, the iRfcb test dataset (48158716) from Torstensson et al. (2024) is used.
<code>max_retries</code>	The maximum number of retry attempts in case of download failure. Default is 3.
<code>sleep_time</code>	The sleep time between download attempts, in seconds. Default is 10.
<code>keep_zip</code>	A logical indicating whether to keep the downloaded zip archive after extraction. Default is FALSE.
<code>verbose</code>	A logical indicating whether to print progress messages. Default is TRUE.
<code>expected_checksum</code>	[Deprecated] Optional. The expected MD5 checksum of the downloaded zip file. If not provided, it is automatically looked up from an internal table based on <code>figshare_article</code> .

Value

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

References

Torstensson, Anders; Skjjevik, 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_download_who_i_plankton
      Download and Extract WHOI-Plankton Data
```

Description

This function downloads WHOI-Plankton annotated plankton images (Sosik et al. 2015) for specified years from <https://hdl.handle.net/1912/7341>. The extracted .png data are saved in the specified destination folder.

Usage

```
ifcb_download_who_i_plankton(
  years,
  dest_folder,
  extract_images = TRUE,
  max_retries = 10,
  quiet = FALSE
)
```

Arguments

years	A vector of years (numeric or character) indicating which datasets to download. The available years are currently 2006 to 2014.
dest_folder	A string specifying the destination folder where the files will be extracted.
extract_images	Logical. If TRUE, extracts .png images from the downloaded archives and removes the .zip files. If FALSE, only downloads the archives without extracting images. Default is TRUE.
max_retries	An integer specifying the maximum number of attempts to retrieve data. Default is 10.
quiet	Logical. If TRUE, suppresses messages about the progress and completion of the download process. Default is FALSE.

Value

If `extract_images = FALSE`, returns a data frame containing metadata of downloaded image files. Otherwise, no return value; files are downloaded and extracted to `dest_folder`.

References

Sosik, H. M., Peacock, E. E. and Brownlee E. F. (2015), Annotated Plankton Images - Data Set for Developing and Evaluating Classification Methods. [doi:10.1575/1912/7341](https://doi.org/10.1575/1912/7341)

Examples

```
## Not run:  
# Download and extract images for 2006 and 2007 in the data folder  
ifcb_download_whoiplankton(c(2006, 2007),  
                           "data",  
                           extract_images = TRUE)  
  
## 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,  
  scale_bar_um = NULL,  
  scale_micron_factor = 1/3.4,  
  scale_bar_position = "bottomright",  
  scale_bar_color = "black",
```

```

old_adc = FALSE,
use_python = FALSE,
gamma = 1,
normalize = FALSE,
add_trailing_numbers = TRUE,
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 .png extraction. Default is NA (include all classes).
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_folders (if provided). Default is TRUE.
overwrite	A logical value indicating whether to overwrite existing PNG files. Default is FALSE.
scale_bar_um	An optional numeric value specifying the length of the scale bar in micrometers. If NULL, no scale bar is added.
scale_micron_factor	A numeric value defining the conversion factor from micrometers to pixels. Defaults to 1/3.4.
scale_bar_position	A character string specifying the position of the scale bar in the image. Options are "topright", "topleft", "bottomright", or "bottomleft". Defaults to "bottomright".
scale_bar_color	A character string specifying the scale bar color. Options are "black" or "white". Defaults to "black".
old_adc	[Deprecated] Previously used to indicate old ADC format. ADC format is now auto-detected from the HDR file and column count. This parameter is ignored.
use_python	Logical. If TRUE, attempts to read the .mat file using a Python-based method. Default is FALSE.
gamma	A numeric value for gamma correction applied to the image. Default is 1 (no correction). Values <1 brighten dark regions, while values >1 darken the image.

normalize	A logical value indicating whether to apply min-max normalization to stretch pixel values to the full 0-255 range. Default is FALSE, preserving raw pixel values comparable to IFCB Dashboard output. See <code>ifcb_extract_pngs()</code> for details.
add_trailing_numbers	Logical. If TRUE, appends a zero-padded numeric suffix derived from the manual class index to the class name when naming output files. If FALSE, uses only the class name without a numeric suffix. Default is TRUE.
roi_folder	[Deprecated] Use <code>roi_folders</code> 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 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

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 # Skip "unclassified"
)

## 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,
  class_files = NULL,
  custom_images = NULL,
  custom_classes = NULL,
  class2use_file = NULL,
  micron_factor = 1/3.4,
  diatom_class = "Bacillariophyceae",
  diatom_include = NULL,
  marine_only = FALSE,
  threshold = "opt",
  multiblob = FALSE,
  feature_recursive = TRUE,
  class_recursive = TRUE,
  drop_zero_volume = FALSE,
  feature_version = NULL,
  use_python = FALSE,
  verbose = TRUE,
  mat_folder = deprecated(),
  mat_files = deprecated(),
  mat_recursive = deprecated()
)
```

Arguments

feature_files	A path to a folder containing feature files or a character vector of file paths.
class_files	(Optional) A character vector of full paths to classification or manual annotation files (.mat, .h5, or .csv), or a single path to a folder containing such files.
custom_images	(Optional) A character vector of image filenames in the format DYYYYMMD-DTHHMMSS_IFCBXXX_ZZZZZ(.png), where "XXX" represents the IFCB number and "ZZZZZ" represents the ROI number. These filenames should

	match the <code>roi_number</code> assignment in the <code>feature_files</code> and can be used as a substitute for classification files.
<code>custom_classes</code>	(Optional) A character vector of corresponding class labels for <code>custom_images</code> .
<code>class2use_file</code>	(Optional) A character string specifying the path to the file containing the <code>class2use</code> variable. Only required for manual results (default: NULL).
<code>micron_factor</code>	Conversion factor from microns per pixel (default: 1/3.4).
<code>diatom_class</code>	A character vector specifying diatom class names in WoRMS. Default: "Bacillariophyceae".
<code>diatom_include</code>	Optional character vector of class names that should always be treated as diatoms, overriding the boolean result of <code>ifcb_is_diatom</code> . Default: NULL.
<code>marine_only</code>	Logical. If TRUE, restricts the WoRMS search to marine taxa only. Default: FALSE.
<code>threshold</code>	A character string controlling which classification to use. "opt" (default) uses the threshold-applied classification, where predictions below the per-class optimal threshold are labeled "unclassified". Any other value (e.g. "all") uses the raw winning class without any threshold applied.
<code>multiblob</code>	Logical. If TRUE, includes multiblob features. Default: FALSE.
<code>feature_recursive</code>	Logical. If TRUE, searches recursively for feature files when <code>feature_files</code> is a folder. Default: TRUE.
<code>class_recursive</code>	Logical. If TRUE and <code>class_files</code> is a folder, searches recursively for classification files. Default: TRUE.
<code>drop_zero_volume</code>	Logical. If TRUE, rows where Biovolume equals zero (e.g., artifacts such as smudges on the flow cell) are removed. Default: FALSE.
<code>feature_version</code>	Optional numeric or character version to filter feature files by (e.g. 2 for "_v2"). Default is NULL (no filtering).
<code>use_python</code>	Logical. If TRUE, attempts to read .mat files using a Python-based method (SciPy). Default: FALSE.
<code>verbose</code>	Logical. If TRUE, prints progress messages. Default: TRUE.
<code>mat_folder</code>	[Deprecated] Use <code>class_files</code> instead.
<code>mat_files</code>	[Deprecated] Use <code>class_files</code> instead.
<code>mat_recursive</code>	[Deprecated] Use <code>class_recursive</code> instead.

Details

- **Classification Data Handling:**

- If `class_files` is provided, the function reads class annotations from .mat, .h5, or .csv files.
- If `custom_images` and `custom_classes` are supplied, they override classification file data (e.g. data from a CNN model).
- If both `class_files` and `custom_images/custom_classes` are given, `class_files` takes precedence.


```
print(biovolume_df_custom)

## End(Not run)
```

```
ifcb_extract_classified_images
```

Extract Taxa Images from Classified Sample

Description

This function reads a classified sample file (.mat, .h5, or .csv) and extracts specified taxa images from the corresponding ROI files, saving each image in a specified directory. Supports .mat files generated by `start_classify_batch_user_training` from the ifcb-analysis repository (Sosik and Olson 2007), .h5 files in IFCB Dashboard `class_scores` format, and .csv files in ClassiPyR-compatible format.

Usage

```
ifcb_extract_classified_images(  
  sample,  
  classified_folder,  
  roi_folder,  
  out_folder,  
  taxa = "All",  
  threshold = "opt",  
  overwrite = FALSE,  
  scale_bar_um = NULL,  
  scale_micron_factor = 1/3.4,  
  scale_bar_position = "bottomright",  
  scale_bar_color = "black",  
  old_adc = FALSE,  
  gamma = 1,  
  normalize = FALSE,  
  use_python = FALSE,  
  verbose = TRUE  
)
```

Arguments

<code>sample</code>	A character string specifying the sample name.
<code>classified_folder</code>	A character string specifying the directory containing the classified files.
<code>roi_folder</code>	A character string specifying the directory containing the ROI files.
<code>out_folder</code>	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.
scale_bar_um	An optional numeric value specifying the length of the scale bar in micrometers. If NULL, no scale bar is added.
scale_micron_factor	A numeric value defining the conversion factor from micrometers to pixels. Defaults to 1/3.4.
scale_bar_position	A character string specifying the position of the scale bar in the image. Options are "topright", "topleft", "bottomright", or "bottomleft". Defaults to "bottomright".
scale_bar_color	A character string specifying the scale bar color. Options are "black" or "white". Defaults to "black".
old_adc	[Deprecated] Previously used to indicate old ADC format. ADC format is now auto-detected from the HDR file and column count. This parameter is ignored.
gamma	A numeric value for gamma correction applied to the image. Default is 1 (no correction). Values <1 brighten dark regions, while values >1 darken the image.
normalize	A logical value indicating whether to apply min-max normalization to stretch pixel values to the full 0-255 range. Default is FALSE, preserving raw pixel values comparable to IFCB Dashboard output. See <code>ifcb_extract_pngs()</code> for details.
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. Additionally, a scale bar can be added to the extracted images based on a specified micron-to-pixel conversion factor.

Usage

```
ifcb_extract_pngs(
  roi_file,
  out_folder = dirname(roi_file),
  ROInumbers = NULL,
  taxaname = NULL,
  gamma = 1,
  normalize = FALSE,
  overwrite = FALSE,
  scale_bar_um = NULL,
  scale_micron_factor = 1/3.4,
  scale_bar_position = "bottomright",
  scale_bar_color = "black",
  old_adc = FALSE,
  verbose = TRUE
)
```

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 brighten dark regions, while values >1 darken the image.
normalize	A logical value indicating whether to apply min-max normalization to stretch pixel values to the full 0-255 range. Default is FALSE, which preserves raw pixel values from the camera, producing images comparable to IFCB Dashboard and other standard IFCB software. Set to TRUE to stretch contrast to the full 0-255 range.
overwrite	A logical value indicating whether to overwrite existing PNG files. Default is FALSE.
scale_bar_um	An optional numeric value specifying the length of the scale bar in micrometers. If NULL, no scale bar is added.
scale_micron_factor	A numeric value defining the conversion factor from micrometers to pixels. Defaults to 1/3.4.
scale_bar_position	A character string specifying the position of the scale bar in the image. Options are "topright", "topleft", "bottomright", or "bottomleft". Defaults to "bottomright".
scale_bar_color	A character string specifying the scale bar color. Options are "black" or "white". Defaults to "black".
old_adc	[Deprecated] Previously used to indicate old ADC format. ADC format is now auto-detected from the HDR file and column count. This parameter is ignored.
verbose	A logical value indicating whether to print progress messages. Default is TRUE.

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))

# Extract images with a 5 micrometer scale bar
ifcb_extract_pngs("path/to/your_roi_file.roi", scale_bar_um = 5)

## 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: <ul style="list-style-type: none">"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 be 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
print(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",
  timestamp_param = "38059",
  max_time_diff_min = 1
)
```

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".

`timestamp_param`
A string specifying the header name for the timestamp column in the Ferrybox data. Default is "38059".

`max_time_diff_min`
Numeric. Maximum allowed difference (in minutes) between the requested timestamp and the closest available Ferrybox data. Defaults to 1 minutes. Timestamps further away than this threshold will not be used for filling missing data.

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

<code>mat_file</code>	A character string specifying the path to the .mat file.
<code>use_python</code>	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

```
# Example .mat file included in the package
mat_file <- system.file("exdata/example.mat", package = "iRfcb")

# Get variable names from a MAT file
variables <- ifcb_get_mat_names(mat_file)
print(variables)
```

`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) <https://github.com/hsosik/ifcb-analysis>

Examples

```
# Example .mat file included in the package  
mat_file <- system.file("exdata/example.mat", package = "ifcb")  
  
# Get class names from a class2use file  
classifier_name <- ifcb_get_mat_variable(mat_file, "classifierName")  
print(classifier_name)
```

```
# Get class names from a classifier file
class2useTB <- ifcb_get_mat_variable(mat_file, "class2useTB")
print(class2useTB)
```

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/en/>.

Usage

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

Arguments

<code>minimal</code>	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/en/>.

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
print(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 quattrosolina",
              "Noctiluca",
              "Gymnodiniales")

ifcb_get_trophic_type(taxa_list)
```

ifcb_is_diatom	<i>Identify Diatoms in Taxa List</i>
----------------	--------------------------------------

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",
  diatom_include = NULL,
  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".
diatom_include	Optional character vector of taxa (or genera) that should always be treated as diatoms, overriding the WoRMS-based classification. Default is NULL.
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

```
# Example taxa
taxa_list <- c("Nitzschia_sp", "Chaetoceros_sp", "Dinophysis_norvegica", "Thalassiosira_sp")

res <- ifcb_is_diatom(taxa_list)
print(res)
```

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 land polygon shape file. The Natural Earth 1:10m land vectors are included as a default shapefile in iRfcb.

Usage

```

ifcb_is_near_land(
  latitudes,
  longitudes,
  distance = 500,
  shape = NULL,
  source = "ne",
  crs = 4326,
  remove_small_islands = TRUE,
  small_island_threshold = 2e+06,
  plot = FALSE,
  verbose = TRUE,
  utm_zone = deprecated()
)

```

Arguments

latitudes	Numeric vector of latitudes for positions.
longitudes	Numeric vector of longitudes for positions. Must be the same length as latitudes.
distance	Buffer distance (in meters) from the coastline to consider "near land." Default is 500 meters.
shape	Optional path to a spatial file (.shp shapefile or .gpkg GeoPackage) containing coastline data. If provided, this file will be used instead of the default Natural Earth 1:10m land vectors. A high-resolution shapefile can improve the accuracy of buffer distance calculations. Alternatively, you can retrieve a more detailed European coastline automatically by setting the source argument to "eea".
source	Character string indicating which default coastline source to use when shape = NULL. Options are "ne" (Natural Earth, default) and "eea" (European Environment Agency, 2017). Ignored if shape is provided.
crs	Coordinate reference system (CRS) to use for input and output. Default is EPSG code 4326 (WGS84).
remove_small_islands	Logical indicating whether to remove small islands from the coastline. Useful in archipelagos. 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 2,000,000 (2 km ²).
plot	A logical indicating whether to plot the points, land polygon and buffer. Default is FALSE.
verbose	A logical indicating whether to print progress messages. Default is TRUE.
utm_zone	[Deprecated] This argument is deprecated. UTM zones are now determined automatically based on the longitude of the input positions.

Details

This function calculates a buffered area around the coastline using a polygon shapefile and determines if each input position intersects with this buffer or the landmass itself. By default, it uses the Natural Earth 1:10m land vector dataset.

The EEA shapefile is downloaded when `source = "eea"` (European Environment Agency, 2017). The downloaded file is cached within an R session.

Value

If `plot = FALSE` (default), a logical vector is returned indicating whether each position is near land or not, with NA for positions where coordinates are missing. If `plot = TRUE`, a `ggplot` object is returned showing the land polygon, buffer area, and position points colored by their proximity to land.

References

European Environment Agency (2017). EEA coastline for analysis (polygon) - version 3.0, March 2017. <https://sdi.eea.europa.eu/catalogue/geoss/api/records/9faa6ea1-372a-4826-a3c7-fb5b05e31c52>

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_list_dashboard_bins

Download bin list from the IFCB Dashboard API

Description

Download bin list from the IFCB Dashboard API

Usage

```
ifcb_list_dashboard_bins(base_url, quiet = FALSE)
```

Arguments

<code>base_url</code>	Character. Base URL to the IFCB Dashboard (e.g. "https://ifcb-data.whoi.edu/").
<code>quiet</code>	Logical. If TRUE, suppresses progress messages. Default is FALSE.

Value

A data frame containing the bin list returned by the API.

See Also

[ifcb_download_dashboard_data\(\)](#) to download data from the IFCB Dashboard API.

[ifcb_download_dashboard_metadata\(\)](#) to retrieve metadata from the IFCB Dashboard API.

Examples

```
bins <- ifcb_list_dashboard_bins("https://ifcb-data.whoi.edu/")
head(bins)
```

ifcb_match_taxa_names *Retrieve WoRMS Records with Retry Mechanism*

Description**[Superseded]**

This function has been superseded by SHARK4R: `match_worms_taxa()` or `worms:wm_records_names()`. It will not receive new features, but will continue to receive critical bug fixes as needed.

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. Default is 3.
sleep_time	A numeric value indicating the number of seconds to wait between retry attempts. Default is 10.
marine_only	Logical. If TRUE, restricts the search to marine taxa only. Default is FALSE.
return_list	A logical value indicating whether 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

```
# Example: Retrieve WoRMS records for a list of taxa names
taxa <- c("Calanus finmarchicus", "Thalassiosira pseudonana", "Phaeodactylum tricornutum")

# Call the function
records <- ifcb_match_taxa_names(taxa_names = taxa,
                               max_retries = 3,
                               sleep_time = 5,
                               marine_only = TRUE,
                               verbose = TRUE)

# Print records as tibble
print(records)
```

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

Python must be installed to use this function. 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 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_prepare_who_i_plankton

Download and Prepare WHOI-Plankton Data

Description

This function downloads manually annotated images from the WHOI-Plankton dataset (Sosik et al. 2015) and generates manual classification files in .mat format that can be used to train an image classifier using the ifcb-analysis MATLAB package (Sosik and Olson 2007).

Usage

```

ifcb_prepare_who_i_plankton(
  years,
  png_folder,
  raw_folder,
  manual_folder,
  class2use_file,
  skip_classes = NULL,
  include_classes = NULL,
  dashboard_url = "https://ifcb-data.who_i.edu/mvco/",
  extract_images = FALSE,
  download_blobs = FALSE,
  blobs_folder = NULL,
  download_features = FALSE,
  features_folder = NULL,
  parallel_downloads = 5,
  sleep_time = 2,
  multi_timeout = 120,
  convert_filenames = TRUE,
  convert_adc = TRUE,
  quiet = FALSE
)

```

Arguments

years	Character vector. Years to download and process. For available years, see https://hdl.handle.net/1912/7341 or ifcb_download_who_i_plankton .
png_folder	Character. Directory where .png images will be stored.
raw_folder	Character. Directory where raw files (.adc, .hdr, .roi) will be stored.
manual_folder	Character. Directory where manual classification files (.mat) will be stored.
class2use_file	Character. File path to .mat file to store the list of available classes.
skip_classes	Character vector. Classes to be excluded during processing. For example images, refer to https://whoigit.github.io/who_i-plankton/ .
include_classes	Character vector. If provided, only these classes will be included during processing. Applied before skip_classes. For example images, refer to https://whoigit.github.io/who_i-plankton/ .
dashboard_url	Character. URL for the IFCB dashboard data source (default: "https://ifcb-data.who_i.edu/mvco/").
extract_images	Logical. If TRUE, extracts .png images from the downloaded archives and removes the .zip files. If FALSE, only downloads the archives without extracting images. Default is FALSE.
download_blobs	Logical. Whether to download blob files (default: FALSE).
blobs_folder	Character. Directory where blob files will be stored (required if download_blobs = TRUE).

download_features	Logical. Whether to download feature files (default: FALSE).
features_folder	Character. Directory where feature files will be stored (required if download_features = TRUE).
parallel_downloads	Integer. Number of parallel IFCB Dashboard downloads (default: 5).
sleep_time	Numeric. Seconds to wait between download requests (default: 2).
multi_timeout	Numeric. Timeout for multiple requests in seconds (default: 120).
convert_filenames	Logical. If TRUE (default), converts filenames of the old format "IFCBxxx_YYYY_DDD_HHMSS" to the new format (DYYYYMMDDTHHMSS_IFCBXXX). [Experimental]
convert_adc	Logical. If TRUE (default), adjusts .adc files from older IFCB instruments (IFCB1–6, with filenames in the format "IFCBxxx_YYYY_DDD_HHMSS") by inserting four empty columns after column 7 to match the newer format. [Experimental]
quiet	Logical. Suppress messages if TRUE (default: 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()`.

This is a wrapper function for the `ifcb_download_who_i_plankton`, `ifcb_download_dashboard_data` and `ifcb_create_manual_file` functions and used for downloading, processing, and converting IFCB data. Please note that this function downloads and extracts large amounts of data, which can take considerable time.

The training data prepared from this function can be merged with an existing training dataset using the `ifcb_merge_manual` function.

Classes included in the training dataset can be controlled using the `include_classes` and `skip_classes` arguments. If `include_classes` is provided, only the specified classes will be processed and included in the output. The `skip_classes` argument can be used to explicitly exclude one or more classes. If both arguments are supplied, `include_classes` is applied first and `skip_classes` is applied afterward.

To exclude individual images rather than entire classes, set `extract_images = TRUE`, manually delete specific .png files from the `png_folder`, and rerun `ifcb_prepare_who_i_plankton`.

If `convert_filenames = TRUE` **[Experimental]**, filenames in the "IFCBxxx_YYYY_DDD_HHMSS" format (used by IFCB1-6) will be converted to IYYYYMMDDTHHMSS_IFCBXXX, ensuring compatibility with blob extraction in `ifcb-analysis` (Sosik & Olson, 2007), which identified the old .adc format by the first letter of the filename.

If `convert_adc = TRUE` **[Experimental]** and `convert_filenames = TRUE` **[Experimental]**, the "IFCBxxx_YYYY_DDD_HHMSS" format will instead be converted to DYYYYMMDDTHHMSS_IFCBXXX. Additionally, .adc files will be modified to include four empty columns (PMT-A peak, PMT-B peak, PMT-C peak, and PMT-D peak), aligning them with the structure of modern .adc files for full compatibility with `ifcb-analysis`.

Value

This function does not return a value but downloads, processes, and stores IFCB data.

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.

Sosik, H. M., Peacock, E. E. and Brownlee E. F. (2015), Annotated Plankton Images - Data Set for Developing and Evaluating Classification Methods. [doi:10.1575/1912/7341](https://doi.org/10.1575/1912/7341)

See Also

<https://hdl.handle.net/1912/7341>, https://whoigit.github.io/whoi-plankton/ifcb_merge_manual
[ifcb_download_whoiplankton](#) [ifcb_download_dashboard_data](#)

Examples

```
## Not run:  
# Download and prepare WHOI-Plankton for the years 2013 and 2014  
ifcb_prepare_whoiplankton(  
  years = c("2013", "2014"),  
  png_folder = "whoi_plankton/png",  
  raw_folder = "whoi_plankton/raw",  
  manual_folder = "whoi_plankton/manual",  
  class2use_file = "whoi_plankton/config/class2use_whoiplankton.mat"  
)  
  
## 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,  
  bins = NULL,  
  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,
    fea_v = 2,
    use_plot_subfolders = TRUE,
    ...
)

```

Arguments

feature_folder	The absolute path to a directory containing all of the feature files for the dataset (version can be defined in fea_v).
hdr_folder	The absolute path to a directory containing all of the hdr files for the dataset.
bins	An optional character vector of bin names (e.g., "D20251021T133007_IFCB134") to restrict processing to a specified subset of bins. If NULL (default), all bins present in feature_folder are processed.
save_data	A logical indicating whether to save data to CSV files. Default is FALSE.
output_file	A string with the base file name for the .csv output (including path). Set to NULL to avoid saving data (default).
plot_folder	The folder where graph images for each sample will be saved. If NULL (default), plots are not saved. If use_plot_subfolders = TRUE, plots are organized into subfolders based on their flag status.
use_marker	A logical 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. Optional.
bubbles	The minimum difference between the starting ESD and the ESD with the most targets. Files with a difference higher than this threshold will be flagged as mostly bubbles. Optional.
incomplete	A numeric vector of length 2 giving the minimum volume of cells (in c/L) and the minimum mL analyzed for a complete run. Files with values below these thresholds will be flagged as incomplete. Optional.
missing_cells	The minimum image count ratio threshold. Files with ratios below this value will be flagged as missing cells. Optional.

biomass	The minimum number of targets in the most populated ESD bin for any given run. Files with fewer targets will be flagged as low biomass. Optional.
bloom	The minimum difference between the starting ESD and the ESD with the most targets. Files with a difference less than this threshold will be flagged as bloom events. This threshold is usually lower than the bubbles threshold. Optional.
humidity	The maximum percent humidity. Files with higher values will be flagged as high humidity. Optional.
micron_factor	Conversion factor from microns per pixel (default: 1/3.4).
fea_v	The version number of the IFCB feature file (e.g., 2, 4). Default is 2, as described in Hayashi et al. 2025. [Experimental]
use_plot_subfolders	A logical indicating whether to save plots in subfolders based on the sample's flag status. If TRUE (default), samples without flags are saved in a "PSD.OK" subfolder, and samples with flags are saved in subfolders named after their flag(s). If FALSE, all plots are saved directly in plot_folder.
...	Additional arguments passed to ggsave(). These override the default width, height, dpi, and background color when saving plots. For example, width = 7, dpi = 300 can be supplied.

Details

The PSD function originates from the PSD Python repository (Hayashi et al. 2025), which can be found at <https://github.com/kudelalab/PSD>.

Python must be installed to use this function. The required Python packages can be installed in a virtual environment using `ifcb_py_install()`.

Value

A list containing three tibbles:

data A tibble with flattened PSD data for each sample.

fits A tibble containing curve fit parameters for each sample.

flags A tibble of flags for each sample, or NULL if no flags are found.

The `save_data` parameter only controls whether CSV files are written to disk; the function always returns this list.

References

Hayashi, K., Enslein, J., Lie, A., Smith, J., Kudela, R.M., 2025. Using particle size distribution (PSD) to automate imaging flow cytobot (IFCB) data quality in coastal California, USA. International Society for the Study of Harmful Algae. <https://doi.org/10.15027/0002041270>

See Also

`ifcb_py_install`, <https://github.com/kudelalab/PSD>

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',
  bins = c("D20211021T133007_IFCB134", "D20211021T140753_IFCB134"),
  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 = 1/2.77,
  fea_v = 2
)

## 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. 2025).

Usage

```
ifcb_psd_plot(sample_name, data, fits, start_fit, flags = NULL)
```

Arguments

<code>sample_name</code>	The name of the sample to plot in <code>DYYYYMMDDTHHMMSS_IFCBXXX</code> .
<code>data</code>	A data frame containing the PSD data (data output from <code>ifcb_psd</code>), where each row represents a sample and each column represents different particle sizes in micrometers.

<code>fits</code>	A data frame containing the fit parameters for the power curve (fits output from <code>ifcb_psd</code>), where each row represents a sample and the columns include the parameters <code>a</code> , <code>k</code> , and <code>R^2</code> .
<code>start_fit</code>	The x-value threshold below which data should be excluded from the plot and fit.
<code>flags</code>	Optional data frame or tibble with columns <code>sample</code> and <code>flag</code> . If <code>sample_name</code> appears in <code>flags\$sample</code> , the corresponding <code>flag</code> text will be displayed on the plot as a red label in the top-left corner.

Value

A ggplot object representing the PSD plot for the sample.

References

Hayashi, K., Enslein, J., Lie, A., Smith, J., Kudela, R.M., 2025. Using particle size distribution (PSD) to automate imaging flow cytobot (IFCB) data quality in coastal California, USA. International Society for the Study of Harmful Algae. <https://doi.org/10.15027/0002041270>

See Also

`ifcb_psd` <https://github.com/kudela1lab/PSD>

Examples

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

# 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
)

# Optional flags
flags <- tibble::tibble(
  sample = "D20230316T101514",
  flag = "Incomplete Run."
)

# Plot PSD of the first sample
plot <- ifcb_psd_plot(
  sample_name = "D20230316T101514",
  data = psd$data,
```

```

    fits = psd$fits,
    start_fit = 10,
    flags = flags
  )

  # 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` (not recommended).

Usage

```

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

```

Arguments

<code>envname</code>	A character string specifying the name of the virtual environment to create. Default is "~/virtualenvs/iRfcb".
<code>use_venv</code>	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.
<code>packages</code>	A character vector of additional Python packages to install. If NULL (default), only the packages from "requirements.txt" are installed.

Details

This function requires Python to be available on the system. It uses the `reticulate` package to manage Python environments and packages.

The `USE_IRFCB_PYTHON` environment variable can be set to automatically activate an installed Python venv named `iRfcb` when the `iRfcb` package is loaded. Ensure that the `iRfcb` venv is installed in `reticulate::virtualenv_root()` and available via `reticulate::virtualenv_list()` (see examples). You can set `USE_IRFCB_PYTHON` to "TRUE" in your `.Renviron` file to enable automatic setup. For more details, see the package README at <https://europeanifcbgroup.github.io/iRfcb/#python-dependency>.

Value

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

Examples

```
## Not run:
# Define the name of the virtual environment in your virtual_root directory
envpath <- file.path(reticulate::virtualenv_root(), "iRfcb")

# Install the iRfcb Python venv in your virtual_root directory
ifcb_py_install(envname = envpath)

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

# Use system Python instead of a virtual environment
ifcb_py_install(envname = envpath, 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,
  feature_version = NULL,
  biovolume_only = 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).

feature_version Optional numeric or character version to filter feature files by (e.g. 2 for "_v2").
Default is NULL (no filtering).

- `biovolume_only` Logical; if TRUE, only a minimal set of feature columns required for biovolume calculations are read from each feature file (typically `roi_number` and `biovolume`). This substantially reduces memory usage and improves performance when other features are not needed. If FALSE, all available feature columns are read.
- `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 only version 4 feature files
v4_features <- ifcb_read_features("path/to/feature_folder", feature_version = 4)

# 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` *Read 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 or character vector specifying the path(s) to .hdr files, or a single 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 timestamps. When `gps_only = TRUE`, only samples with GPS coordinates are included.

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	<i>Read a MATLAB .mat File in R</i>
---------------	-------------------------------------

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

Python must be installed to use this function. The required python packages can be installed in a virtual environment using `ifcb_py_install()`.

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:
# Initialize Python environment and install required packages
ifcb_py_install()

# Example .mat file included in the package
mat_file <- system.file("exdata/example.mat", package = "iRfcb")

# Read mat file using Python
data <- ifcb_read_mat(mat_file)

## 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

```
mat_file <- system.file("exdata/example_summary.mat", package = "iRfcb")

summary_data <- ifcb_read_summary(mat_file, biovolume = FALSE, threshold = "opt")
print(summary_data)
```

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

Python must be installed to use this function. 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

[Deprecated]

This function was deprecated as there is a better alternative: `ClassiPyR::run_app()`. For more information, please see: <https://europeanifcbgroup.github.io/ClassiPyR/>

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

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.

Usage

```
ifcb_run_image_gallery()
```

Value

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

Examples

```
# Run the IFCB image gallery Shiny app
if(interactive()){
  ifcb_run_image_gallery()
}
```

ifcb_save_classification

Classify an IFCB Sample and Save Results

Description

Extracts PNG images from an IFCB .roi file, classifies each image via the Gradio API predict_scores endpoint (returning all class scores), fetches per-class thresholds, and writes the results in the specified format.

Usage

```
ifcb_save_classification(
  roi_file,
  output_folder,
  format = c("h5", "mat", "csv"),
  gradio_url = "https://irfcb-classify.hf.space",
  model_name = "SMHI NIVA ResNet50 V5",
  verbose = TRUE,
  ...
)
```

Arguments

roi_file	A character string specifying the path to the .roi file.
output_folder	A character string specifying the directory where the output file will be saved. The file is named automatically based on the sample name (e.g. D20220522T003051_IFCB134_class.h5, D20220522T003051_IFCB134_class_v1.mat, or D20220522T003051_IFCB134.csv).
format	A character string specifying the output format. One of "h5" (default), "mat", or "csv".
gradio_url	A character string specifying the base URL of the Gradio application. Default is "https://irfcb-classify.hf.space", which is an example Hugging Face Space with limited resources intended for testing and demonstration. For large-scale classification, deploy your own instance of the classification app (source code: https://github.com/EuropeanIFCBGroup/ifcb-inference-app) and pass its URL here.
model_name	A character string specifying the name of the CNN model to use for classification. Default is "SMHI NIVA ResNet50 V5". Use <code>ifcb_classify_models()</code> to list all available models.
verbose	A logical value indicating whether to print progress messages. Default is TRUE.
...	Additional arguments passed to <code>ifcb_extract_pngs()</code> (e.g. ROI numbers, gamma).

Details

Three output formats are supported:

"h5" IFCB Dashboard class_scores v3 HDF5 format. Contains output_scores, class_labels, roi_numbers (Dashboard-required), plus classifier_name, class_name, class_name_auto, and thresholds. Requires the **hdf5r** package.

"mat" IFCB Dashboard class_scores v1 MATLAB format. Contains class2useTB, TBscores, roinum, TBclass, TBclass_above_threshold, and classifierName. Requires Python with **scipy** and **numpy**.

"csv" ClassiPyR-compatible CSV format with columns file_name, class_name (threshold-applied), class_name_auto (winning class without threshold), and score (winning class confidence). See <https://github.com/EuropeanIFCBGroup/ClassiPyR> for details.

Value

The path to the saved file (invisibly).

See Also

[ifcb_classify_images\(\)](#), [ifcb_classify_sample\(\)](#), [ifcb_classify_models\(\)](#)

Examples

```
## Not run:
# Classify a sample and save as HDF5 (default)
ifcb_save_classification(
  "path/to/D20220522T003051_IFCB134.roi",
  output_folder = "output"
)

# Save as Dashboard v1 .mat format
ifcb_save_classification(
  "path/to/D20220522T003051_IFCB134.roi",
  output_folder = "output",
  format = "mat"
)

# Save as CSV
ifcb_save_classification(
  "path/to/D20220522T003051_IFCB134.roi",
  output_folder = "output",
  format = "csv"
)

## 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 classification files (.mat, .h5, or .csv), 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,  
    class_files = NULL,  
    class2use_file = NULL,  
    hdr_folder = NULL,  
    custom_images = NULL,  
    custom_classes = NULL,  
    micron_factor = 1/3.4,  
    diatom_class = "Bacillariophyceae",  
    diatom_include = NULL,  
    marine_only = FALSE,  
    threshold = "opt",  
    feature_recursive = TRUE,  
    class_recursive = TRUE,  
    hdr_recursive = TRUE,  
    drop_zero_volume = FALSE,  
    feature_version = NULL,  
    use_python = FALSE,  
    verbose = TRUE,  
    mat_folder = deprecated(),  
    mat_files = deprecated(),  
    mat_recursive = deprecated()  
)
```

Arguments

- | | |
|-----------------------------|---|
| <code>feature_folder</code> | Path to the folder containing feature files (e.g., CSV format). |
| <code>class_files</code> | (Optional) A character vector of full paths to classification or manual annotation files (.mat, .h5, or .csv), or a single path to a folder containing such files. |
| <code>class2use_file</code> | (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 DYYYYMMD-DTHHMMSS_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 classification 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 character vector of diatom class names in the World Register of Marine Species (WoRMS). Default is "Bacillariophyceae".
diatom_include	Optional character vector of class names that should always be treated as diatoms, overriding the boolean result of ifcb_is_diatom. Default: NULL.
marine_only	Logical. If TRUE, restricts the WoRMS search to marine taxa only. Default is FALSE.
threshold	A character string controlling which classification to use. "opt" (default) uses the threshold-applied classification, where predictions below the per-class optimal threshold are labeled "unclassified". Any other value (e.g. "all") uses the raw winning class without any threshold applied.
feature_recursive	Logical. If TRUE, the function will search for feature files recursively within the feature_folder. Default is TRUE.
class_recursive	Logical. If TRUE, the function will search for classification files recursively when class_files is a 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.
drop_zero_volume	Logical. If TRUE, rows where Biovolume equals zero (e.g., artifacts such as smudges on the flow cell) are removed. Default: FALSE.
feature_version	Optional numeric or character version to filter feature files by (e.g. 2 for "_v2"). Default is NULL (no filtering).
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.
mat_folder	[Deprecated] Use class_files instead.
mat_files	[Deprecated] Use class_files instead.
mat_recursive	[Deprecated] Use class_recursive instead.

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 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 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*, 45(3), 569-579, 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/classified",
                          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

<code>classpath_generic</code>	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/').
<code>hdr_folder</code>	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.
<code>year_range</code>	Numeric vector specifying the range of years (e.g., 2013:2014) to process.
<code>use_python</code>	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.
m1_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('path/to/class/classxxx_v1/',
                           'path/to/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,  
  feature_version = 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).

`feature_version` Optional numeric or character version to filter feature files by (e.g. 2 for "_v2"). Default is NULL (no filtering).

`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 inhibit time, 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 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 inhibit time, 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.
- **inhibit_time**: A numeric vector of inhibit time 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.adc"  
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

<code>latitudes</code>	A numeric vector of latitude points.
<code>longitudes</code>	A numeric vector of longitude points.
<code>plot</code>	A boolean indicating whether to plot the points along with the sea basins. Default is <code>FALSE</code> .
<code>shape_file</code>	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 <code>iRfcb</code> 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/en/>). 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_images_by_class`*Zip Image Subfolders by Class*

Description

This function creates one zip archive per immediate subdirectory in a folder containing image files. Each archive corresponds to a single class or taxon.

Usage

```
ifcb_zip_images_by_class(  
  image_folder,  
  output_dir,  
  n_images = NULL,  
  quiet = FALSE  
)
```

Arguments

<code>image_folder</code>	The directory containing subdirectories with image files.
<code>output_dir</code>	The directory where the zip archives will be written.
<code>n_images</code>	Integer. Maximum number of images to randomly sample per subdirectory. If NULL, all images are included.
<code>quiet</code>	Logical. If TRUE, suppresses the progress bar. Default is FALSE.

Details

When `n_images` is specified, images are randomly sampled without replacement from each subdirectory. When `n_images` is NULL, all images in each subdirectory are included.

Supported image formats (case-insensitive) are: png, jpg, jpeg, tif, tiff, bmp, and gif.

Value

This function does not return any value; it creates one zip archive per subdirectory containing images.

Examples

```
## Not run:  
# Set a random seed to reproduce the random sampling  
set.seed(123)  
  
# Create zip archives for each subdirectory with up to 50 random images  
ifcb_zip_images_by_class(  
  image_folder = "path/to/images",  
  output_dir = "path/to/zips",
```

```

    n_images = 50
)

## End(Not run)

```

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](https://doi.org/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)
```

process_ifcb_string *Process IFCB String*

Description

This helper function processes IFCB (Imaging FlowCytobot) filenames and extracts the date component in YYYYMMDD format. It supports two formats:

- IFCB1_2014_188_222013: Extracts the date using year and day-of-year information.
- D20240101T120000_IFCB1: Extracts the date directly from the timestamp.

Usage

```
process_ifcb_string(ifcb_string, quiet = FALSE)
```

Arguments

ifcb_string	A character vector of IFCB filenames to process.
quiet	A logical indicating whether to suppress messages for unknown formats. Defaults to FALSE.

Value

A character vector containing extracted dates in YYYYMMDD format, or NA for unknown formats.

Examples

```
# Example 1: Process a string in the 'IFCB1_2014_188_222013' format
process_ifcb_string("IFCB1_2014_188_222013")

# Example 2: Process a string in the 'D20240101T120000_IFCB1' format
process_ifcb_string("D20240101T120000_IFCB1")

# Example 3: Process an unknown format
process_ifcb_string("UnknownFormat_12345")
```

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	<i>Split Large Zip File into Smaller Parts</i>
-----------------	--

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 or .h5 format) and summarizes the number of targets in each class based on the classification scores and thresholds.

Usage

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

Arguments

classfile	Character string specifying the path to the classifier result file (.mat or .h5 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. Not available for .h5 files.
use_python	Logical. If TRUE, uses Python-based reading for .mat files. Default is FALSE.

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 microns³ to carbon in picograms for large diatoms (> 2000 micron³) according to Menden-Deuer and Lessard 2000. The formula used is: $\log \text{pgC cell}^{-1} = \log a + b * \log V \text{ (um}^3\text{)}$, with $\log a = -0.933$ and $b = 0.881$ for diatoms > 3000 um³.

Usage

```
vol2C_lgdiatom(volume)
```

Arguments

volume A numeric vector of biovolume measurements in microns³.

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 microns³ to carbon in picograms for protists besides large diatoms (> 3000 micron³) according to Menden-Deuer and Lessard 2000. The formula used is: $\log \text{pgC cell}^{-1} = \log a + b * \log V \text{ (um}^3\text{)}$, with $\log a = -0.665$ and $b = 0.939$.

Usage

```
vol2C_nondiatom(volume)
```

Arguments

volume A numeric vector of biovolume measurements in microns³.

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

create_package_manifest, 3

ifcb_adjust_classes, 4, 19
ifcb_annotate_batch, 5
ifcb_annotate_samples, 7
ifcb_classify_images, 9
ifcb_classify_images(), 11, 13, 70
ifcb_classify_models, 11
ifcb_classify_models(), 10, 12, 13, 69, 70
ifcb_classify_sample, 12
ifcb_classify_sample(), 10, 11, 70
ifcb_convert_filenames, 14, 76
ifcb_correct_annotation, 7, 15
ifcb_count_mat_annotations, 16
ifcb_create_class2use, 5, 9, 18
ifcb_create_manifest, 19
ifcb_create_manual_file, 7, 20, 55
ifcb_download_dashboard_data, 21, 55, 56
ifcb_download_dashboard_data(), 24, 50
ifcb_download_dashboard_metadata, 24
ifcb_download_dashboard_metadata(), 23, 50
ifcb_download_test_data, 25
ifcb_download_who_i_plankton, 26, 54–56
ifcb_extract_annotated_images, 27, 35, 36
ifcb_extract_biovolumes, 30
ifcb_extract_classified_images, 29, 33, 36
ifcb_extract_pngs, 29, 35, 35
ifcb_extract_pngs(), 10, 12, 13, 29, 34, 69
ifcb_get_ecotaxa_example, 37
ifcb_get_ferrybox_data, 38
ifcb_get_mat_names, 39, 41
ifcb_get_mat_variable, 40, 40
ifcb_get_runtime, 42
ifcb_get_shark_colnames, 43, 44
ifcb_get_shark_example, 43, 44
ifcb_get_trophic_type, 44
ifcb_is_diatom, 32, 45
ifcb_is_in_basin, 46
ifcb_is_near_land, 47
ifcb_list_dashboard_bins, 49
ifcb_list_dashboard_bins(), 23, 24
ifcb_match_taxa_names, 50
ifcb_merge_manual, 51, 55, 56
ifcb_prepare_who_i_plankton, 53
ifcb_psd, 56, 60
ifcb_psd_plot, 59
ifcb_py_install, 5, 9, 16, 19, 53, 58, 61, 65, 67
ifcb_read_features, 32, 62
ifcb_read_hdr_data, 63, 76
ifcb_read_mat, 64
ifcb_read_summary, 65
ifcb_replace_mat_values, 66
ifcb_run_image_gallery, 68
ifcb_save_classification, 69
ifcb_summarize_biovolumes, 71
ifcb_summarize_class_counts, 74
ifcb_summarize_png_counts, 75
ifcb_summarize_png_metadata, 77
ifcb_volume_analyzed, 78
ifcb_volume_analyzed_from_adc, 79
ifcb_which_basin, 80
ifcb_zip_images_by_class, 81
ifcb_zip_matlab, 82, 85
ifcb_zip_pngs, 84, 84

process_ifcb_string, 86

read_hdr_file, 86

split_large_zip, 87
summarize_TBclass, 88

vol2C_lgdiatom, 88
vol2C_nondiatom, 89