

Package ‘iRfcb’

August 19, 2026

Type Package

Title Tools for Managing Imaging FlowCytobot (IFCB) Data

Version 0.10.0

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

Suggests hdf5r, knitr, mockery, R.matlab, 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 kudelalab licensed under the MIT License.

Config/Needs/website rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

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 2026-08-19 19:20:02 UTC

Contents

create_package_manifest	4
ifcb_adjust_classes	4
ifcb_annotate_batch	5
ifcb_annotate_samples	8
ifcb_classify_images	10
ifcb_classify_models	11
ifcb_classify_sample	12
ifcb_convert_filenames	14
ifcb_correct_annotation	15
ifcb_count_mat_annotations	17
ifcb_create_class2use	18
ifcb_create_manifest	19
ifcb_create_manual_file	20
ifcb_download_dashboard_data	22
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	34
ifcb_extract_features	37
ifcb_extract_pngs	40
ifcb_get_ecotaxa_example	42
ifcb_get_ferrybox_data	43
ifcb_get_mat_names	44
ifcb_get_mat_variable	46
ifcb_get_runtime	47

ifcb_get_shark_colnames	48
ifcb_get_shark_example	49
ifcb_get_trophic_type	49
ifcb_is_diatom	50
ifcb_is_in_basin	52
ifcb_is_near_land	53
ifcb_list_dashboard_bins	55
ifcb_match_taxa_names	56
ifcb_merge_manual	57
ifcb_prepare_who_i_plankton	59
ifcb_psd	62
ifcb_psd_plot	66
ifcb_py_install	67
ifcb_qc_sample	70
ifcb_read_features	72
ifcb_read_hdr_data	74
ifcb_read_mat	75
ifcb_read_summary	76
ifcb_replace_mat_values	77
ifcb_run_image_gallery	78
ifcb_save_classification	79
ifcb_summarize_biovolumes	81
ifcb_summarize_cell_counts	85
ifcb_summarize_class_counts	88
ifcb_summarize_png_counts	89
ifcb_summarize_png_metadata	91
ifcb_volume_analyzed	92
ifcb_volume_analyzed_from_adc	93
ifcb_which_basin	94
ifcb_zip_images_by_class	95
ifcb_zip_matlab	96
ifcb_zip_pngs	98
process_ifcb_string	100
read_hdr_file	100
split_large_zip	101
summarize_TBclass	102
vol2C_diatom	102
vol2C_diatom_auto	103
vol2C_lgdiatom	104
vol2C_nondiatom	105

```
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

<code>paths</code>	A character vector of paths to files and/or directories to include in the manifest.
<code>manifest_path</code>	A character string specifying the path to the manifest file. Default is "MANIFEST.txt".
<code>temp_dir</code>	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

The MAT files are read and written directly from R, producing output identical to the MATLAB ifcb-analysis format.

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_create_class2use` <https://github.com/hsosik/ifcb-analysis>

Examples

```
## Not run:
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, which takes a vector of file paths. This rename applies to this function only, and is not a package-wide rename: <code>ifcb_annotate_samples()</code> genuinely takes one directory and keeps adc_folder.

Details

The .mat files are created and edited directly from R.

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 (YYYYMMDD) 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:
# 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

The MAT files are written directly from R.

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_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://ifcb.serve.scilifelab.se",
    top_n = 1,
    model_name = "SMHI NIVA SYKE SAMS SZN ResNet 50 V6",
    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://ifcb.serve.scilifelab.se", an instance hosted on the SciLife-Lab Serve platform. A free example Hugging Face Space is also available at "https://irfcb-classify.hf.space" (limited resources, intended for testing and demonstration). For large-scale or production 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 SYKE SAMS SZN ResNet 50 V6". Use ifcb_classify_models() 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.

Transient network failures (dropped connections and HTTP 429/5xx responses, as seen for example when the hosted server restarts) are retried automatically up to four times with exponential backoff before an image is reported as failed. Set `options(irfcb.gradio_max_tries = )` to change the number of attempts.

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://ifcb.serve.scilifelab.se")
```

Arguments

`gradio_url` A character string specifying the base URL of the Gradio application. Default is "https://ifcb.serve.scilifelab.se", an instance hosted on the SciLife-Lab Serve platform. A free example Hugging Face Space is also available at "https://ifcb-classify.hf.space" (limited resources, intended for testing and demonstration). For large-scale or production 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://ifcb.serve.scilifelab.se",
  top_n = 1,
  model_name = "SMHI NIVA SYKE SAMS SZN ResNet 50 V6",
```

```

    verbose = TRUE,
    ...
)

```

Arguments

<code>roi_file</code>	A character string specifying the path to the <code>.roi</code> file.
<code>gradio_url</code>	A character string specifying the base URL of the Gradio application. Default is <code>"https://ifcb.serve.scilifelab.se"</code> , an instance hosted on the SciLife-Lab Serve platform. A free example Hugging Face Space is also available at <code>"https://irfcb-classify.hf.space"</code> (limited resources, intended for testing and demonstration). For large-scale or production 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 <code>Inf</code> to return all predictions.
<code>model_name</code>	A character string specifying the name of the CNN model to use for classification. Default is <code>"SMHI NIVA SYKE SAMS SZN ResNet 50 V6"</code> . Use <code>ifcb_classify_models()</code> to list all available models.
<code>verbose</code>	A logical value indicating whether to print progress messages. Default is <code>TRUE</code> .
<code>...</code>	Additional arguments passed to <code>ifcb_extract_pngs()</code> (e.g. <code>ROInumbers</code> , <code>gamma</code> , <code>scale_bar_um</code>).

Details

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

Transient network failures (dropped connections and HTTP 429/5xx responses, as seen for example when the hosted server restarts) are retried automatically up to four times with exponential backoff before an image is reported as failed. Set `options(iRfcb.gradio_max_tries = )` to change the number of attempts.

Value

A data frame with the following columns:

<code>file_name</code>	The PNG file name of the classified image.
<code>class_name</code>	The predicted class name with per-class thresholds applied; <code>"unclassified"</code> if the score is below the threshold.
<code>class_name_auto</code>	The winning class name without any threshold applied (argmax of scores).
<code>score</code>	The prediction confidence score (0–1).
<code>model_name</code>	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(filenames, tz = "UTC")
```

Arguments

filenames	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.
tz	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

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

ifcb_correct_annotation

Correct Annotations in MATLAB Classlist Files

Description

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

Usage

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

Arguments

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

Details

The MAT files are read and written directly from R, producing output identical to the MATLAB ifcb-analysis format.

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

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

Examples

```
## Not run:
# Correct class ID in .mat classlist files using a correction file
ifcb_correct_annotation("input/manual",
                        "output/manual",
                        "corrections.txt",
                        99)

# Correct class ID in .mat classlist files using a character vector of filenames
```

```

ifcb_correct_annotation("input/manual",
                        "output/manual",
                        c("D20230917T153755_IFCB134_01724.png",
                          "D20230917T110059_IFCB134_00380.png"),
                        99)

## End(Not run)

```

ifcb_count_mat_annotations

Count IFCB Annotations from .mat Files

Description

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

Usage

```

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

```

Arguments

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

Details

If `use_python = TRUE`, the function tries to read the `.mat` file using `ifcb_read_mat()`, which relies on SciPy. This approach may be faster than the default R reader, 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()`.

Otherwise, the function reads the `.mat` file with the default R reader.

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

The .mat file is written directly from R, producing output identical to the MATLAB ifcb-analysis format.

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

Examples

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

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

## End(Not run)
```

`ifcb_create_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

<code>roi_length</code>	Integer. The number of rows in the class list (number of ROIs).
<code>class2use</code>	Character vector. The names of the classes to include in the <code>class2use_manual</code> field of the MAT file.
<code>output_file</code>	Character. The path where the output MAT file will be saved.
<code>classlist</code>	Integer or numeric vector. Defines the values for the second column of the class list, typically representing the manual classification labels: • 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 <code>roi_length</code> is provided, the corresponding values will be used per row.
<code>do_compression</code>	A logical value indicating whether to compress the .mat file. Default is TRUE.

Details

The MAT file is written directly from R, producing output identical to the MATLAB `ifcb-analysis` format.

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:
# 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.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.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

```
# Requires an internet connection and a reachable dashboard; wrapped in try()
# so the example degrades gracefully when the service is unavailable.
try(
  ifcb_download_dashboard_data(
    dashboard_url = "https://ifcb-data.who.who.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

base_url	Character. Base URL to the IFCB Dashboard (e.g. "https://ifcb-data.who.who.edu/").
dataset_name	Optional character. Dataset slug (e.g. "mvco") to retrieve metadata for a specific dataset. If NULL, all available metadata are downloaded.
quiet	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

```
# Requires an internet connection and a reachable dashboard; wrapped in
# try() so the example degrades gracefully when the service is unavailable.
try({
  # Download metadata for a specific dataset
  metadata_mvco <- ifcb_download_dashboard_metadata("https://ifcb-data.whoi.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

dest_dir	The destination directory where the files will be unzipped.
figshare_article	The file article number at the SciLifeLab Figshare data repository. By default, the iRfcb test dataset (48158716) from Torstensson et al. (2024) is used.
max_retries	The maximum number of retry attempts in case of download failure. Default is 3.
sleep_time	The sleep time between download attempts, in seconds. Default is 10.

keep_zip A logical indicating whether to keep the downloaded zip archive after extraction. Default is FALSE.

verbose A logical indicating whether to print progress messages. Default is TRUE.

expected_checksum **[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 figshare_article.

Value

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

References

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

Examples

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

## End(Not run)
```

ifcb_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_who_i_plankton(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

<code>manual_folder</code>	A character string specifying the path to the directory containing the manually classified .mat files.
<code>class2use_file</code>	A character string specifying the path to the file containing class names.
<code>roi_folders</code>	A character vector specifying one or more directories containing the ROI files.
<code>out_folder</code>	A character string specifying the output directory where the extracted images will be saved.
<code>skip_class</code>	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).
<code>verbose</code>	A logical value indicating whether to print progress messages. Default is TRUE.
<code>manual_recursive</code>	Logical. If TRUE, the function will search for MATLAB files recursively within the <code>manual_folder</code> . Default is FALSE.
<code>roi_recursive</code>	Logical. If TRUE, the function will search for data files recursively within the <code>roi_folders</code> (if provided). Default is TRUE.
<code>overwrite</code>	A logical value indicating whether to overwrite existing PNG files. Default is FALSE.
<code>scale_bar_um</code>	An optional numeric value specifying the length of the scale bar in micrometers. If NULL, no scale bar is added.
<code>scale_micron_factor</code>	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 ifcb_extract_pngs() 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 roi_folders instead.

Details

If `use_python = TRUE`, the function tries to read the .mat file using `ifcb_read_mat()`, which relies on SciPy. This approach may be faster than the default R reader, 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()`.

Otherwise, the function reads the .mat file with the default R reader.

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,
  diatom_equation = c("large", "all", "auto"),
  threshold = "opt",
  multiblob = FALSE,
  feature_recursive = TRUE,
  class_recursive = TRUE,
  drop_zero_volume = FALSE,
  feature_version = NULL,
  use_cell_counts = FALSE,
  single_cell_values = c(-1, 0),
  carbon_conversion = c("roi", "cell"),
```

```

    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. Supply a single file format per sample: a sample represented twice (e.g. by both a .mat and a .h5) would have its ROIs counted once per file, so this is rejected with an error naming the affected samples.
- 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.
- class2use_file** (Optional) A character string specifying the path to the file containing the class2use variable. Only required for manual results (default: NULL).
- micron_factor** Conversion factor from microns per pixel (default: 1/3.4).
- diatom_class** A character vector specifying diatom class names in WoRMS. Default: "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: FALSE.
- diatom_equation** A character string selecting which Menden-Deuer and Lessard (2000) carbon-to-volume relationship to apply to diatoms. "large" (default) uses the large-diatom ($> 3000 \text{ micron}^3$) equation (vol2C_lgdiatom), matching the ifcb-analysis convention. "all" uses the all-sizes diatom equation (vol2C_diatom), which assigns more carbon to small cells. "auto" (vol2C_diatom_auto) selects between them per volume, using the large-diatom equation above 3000 micron^3 and the all-sizes equation at or below it, so each stays inside its calibrated size range. "auto" needs no cell_count data and is independent of carbon_conversion. Be aware that the two are discontinuous at that boundary (about 190 pgC against 135 pgC), so "auto" makes predicted carbon fall as a cell grows across it; that is why it is not the default. "auto" selects on whatever volume the conversion receives, which is the per-cell volume when carbon_conversion = "cell". Note that biovolume is measured per region of interest (ROI/image), so it is not a per-cell volume unless carbon_conversion = "cell" is used: chains of small cells register a large ROI biovolume. Non-diatom protists always use vol2C_nondiatom regardless of this setting.

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.
multiblob	Logical. If TRUE, includes multiblob features. Default: FALSE.
feature_recursive	Logical. If TRUE, searches recursively for feature files when feature_files is a folder. Default: TRUE.
class_recursive	Logical. If TRUE and class_files is a folder, searches recursively for classification files. Default: 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_cell_counts	Logical. If TRUE, reads the optional per-ROI cell_count data stored by the diatom chain counter in .mat/.h5/.csv classification files and adds cell_count (raw) and cell_count_resolved (resolved abundance) columns to the output. Only supported with automated class_files; not with manual files or custom_images. Default: FALSE.
single_cell_values	Integer vector of cell_count values that should be treated as a single cell when resolving cell_count_resolved. Default is c(-1, 0), i.e. ROIs that were not counted (-1) and ROIs where no cells were detected (0) each count as one cell. Values not listed are used verbatim. Only used when use_cell_counts = TRUE.
carbon_conversion	A character string controlling how the Menden-Deuer and Lessard (2000) relationships are applied. "roi" (default) applies the selected equation once to the whole ROI biovolume, matching the ifcb-analysis convention and reproducing previous results exactly. "cell" divides the ROI biovolume by the resolved cell count, applies the equation to that per-cell volume, and multiplies back by the count, which is how the relationships are defined ($\log \text{ pgC cell}^{-1}$). Requires use_cell_counts = TRUE. carbon_pg remains a per-ROI total in both cases, not carbon per cell. ROIs holding a single cell, ROIs that were never chain-counted, and ROIs from files carrying no cell_count data are all converted as one cell, so only ROIs with cell_count ≥ 2 change.
use_python	Logical. If TRUE, attempts to read .mat files using a Python-based method (SciPy). Default: FALSE.
verbose	Logical. If TRUE, prints progress messages. Default: TRUE.
mat_folder	[Deprecated] Use class_files instead.
mat_files	[Deprecated] Use class_files instead.
mat_recursive	[Deprecated] Use class_recursive 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.
- **MAT File Processing:**
 - If `use_python = TRUE`, the function reads `.mat` files using `ifcb_read_mat()` (requires Python + SciPy).
 - Otherwise, it reads `.mat` files with the default R reader.
- **Per-cell carbon conversion:**
 - The Menden-Deuer and Lessard (2000) relationships are fitted per cell ($\log \text{pgC cell}^{-1} = \log a + b * \log V$), but an IFCB biovolume describes a whole region of interest, which for a chain-forming diatom is the whole chain. Every one of these relationships has $b < 1$, so applying one to an aggregated chain volume returns less carbon than applying it per cell and summing. The two differ by a factor of $n^{(1-b)}$: about 1.28 for an 8-cell chain and 1.43 for 20 cells under the large-diatom equation.
 - `carbon_conversion = "cell"` divides the ROI biovolume evenly among the counted cells, which assumes the cells in a chain are of similar size.
 - It further assumes the ROI biovolume is cell volume. This is weakest for *Chaetoceros*, whose setae add to the measured ROI biovolume without being cell material, so the per-cell volume is overestimated and per-cell carbon is biased high; whole-ROI conversion biases it low instead. Neither is corrected here.

Value

A data frame containing:

- `sample`: The sample name.
- `classifier`: The classifier used (if applicable).
- `roi_number`: The region of interest (ROI) number.
- `class`: The identified taxonomic class.
- `biovolume_um3`: Computed biovolume in cubic micrometers.
- `carbon_pg`: Estimated carbon content in picograms.
- `cell_count`, `cell_count_resolved` (only when `use_cell_counts = TRUE`): the raw per-ROI cell count and the resolved number of cells used for abundance.

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.

Groves, G. J. J., Arthur, G., Bresnan, E., Whyte, C., Arce, P. and Davidson, K. (2026), Automatic enumeration of chains of marine diatoms using "You Only Look Once" - a machine learning approach. *Journal of Plankton Research*, 48(2), fbaf064, doi: 10.1093/plankt/fbaf064.

See Also

[ifcb_read_features](#) [ifcb_is_diatom](#) [ifcb_summarize_cell_counts](#) <https://github.com/nodc-sweden/ifcb-pytorch-classify> <https://www.marinespecies.org/>

Examples

```
## Not run:
# Using classification results:
feature_files <- "data/features"
class_files <- "data/classified"

biovolume_df <- ifcb_extract_biovolumes(feature_files,
                                       class_files)

print(biovolume_df)

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

biovolume_df_custom <- ifcb_extract_biovolumes(feature_files,
                                              custom_images = images,
                                              custom_classes = classes)

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

sample	A character string specifying the sample name.
classified_folder	A character string specifying the directory containing the classified files.
roi_folder	A character string specifying the directory containing the ROI files.
out_folder	A character string specifying the directory to save the extracted images.
taxa	A character string specifying the taxa to extract. Default is "All".
threshold	A character string specifying the threshold to use ("none", "opt", "adhoc"). Default is "opt".
overwrite	A logical value indicating whether to overwrite existing PNG files. Default is FALSE.
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.

<code>gamma</code>	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.
<code>normalize</code>	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 ifcb_extract_pngs() for details.
<code>use_python</code>	Logical. If TRUE, attempts to read the .mat file using a Python-based method. Default is FALSE.
<code>verbose</code>	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 R reader, 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()`.

Otherwise, the function reads the .mat file with the default R reader.

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_features *Extract Slim Features and Blobs from IFCB Data*

Description

This function computes the "slim" feature set (version 4) and blob masks from raw Imaging Flow-Cytobot (IFCB) data by calling the WHOI ifcb-features Python package. For each bin it writes a feature table (<bin>_features_v4.csv, 30 morphological features per region of interest) and an archive of binary blob masks (<bin>_blobs_v4.zip, one 1-bit PNG per ROI). Features and blobs are written to separate, user-specified directories.

Usage

```
ifcb_extract_features(
    data_folder,
    features_folder,
    blobs_folder,
    bins = NULL,
    parallel = FALSE,
    n_cores = NULL,
    overwrite = FALSE,
    feature_tag = c("features", "fea"),
    multiblob = FALSE,
    backend = NULL,
    verbose = TRUE
)
```

Arguments

data_folder	The path to a directory containing raw IFCB data (.roi, .adc and .hdr files). The directory is searched recursively by the raw-data reader, so nested data structures are supported.
features_folder	The path to the directory where the <bin>_features_v4.csv files will be written. Created if it does not exist.
blobs_folder	The path to the directory where the <bin>_blobs_v4.zip files will be written. Created if it does not exist.
bins	An optional character vector of bin names (e.g. "D20220522T003051_IFCB134") to restrict processing to a subset of bins. If NULL (default), all bins found in data_folder are processed.
parallel	A logical indicating whether to process bins in parallel. Default is FALSE.
n_cores	An integer specifying the number of parallel workers to use when parallel = TRUE (worker processes on Linux, threads on Windows and macOS; see Details). If NULL (default), parallel::detectCores() - 1 workers are used. Ignored when parallel = FALSE.

overwrite	A logical indicating whether to overwrite existing feature and blob files. If FALSE (default), bins whose outputs already exist are skipped.
feature_tag	A string controlling the token between the bin lid and the version in the feature file name. "features" (default) writes <bin>_features_v4.csv (the upstream ifcb-features convention); "fea" writes <bin>_fea_v4.csv, the name the IFCB Dashboard (ifcbdb / pyifcb's FeaturesDirectory) searches for. Use "fea" when the output is destined for an IFCB Dashboard instance; remember the dataset directory there must be registered with product version 4 to match the _v4 suffix. The blob archive name (<bin>_blobs_v4.zip) is unaffected.
multiblob	A logical indicating whether to additionally write multiblob/<bin>_multiblob_v4.csv files (per-blob features for regions of interest with more than one blob) inside features_folder. Bins without multi-blob ROIs get no sidecar file, as in upstream ifcb-features. Requires ifcb-features v1.2.0 or later; see Details. Default is FALSE.
backend	An optional string forcing the raw-data reader, either "ifcbkit" or "pyifcb". If NULL (default), the IRFCB_IFCB_BACKEND environment variable is used when set, otherwise the preferred available reader (ifcbkit when both are installed). See Details for the cases in which the two readers differ.
verbose	A logical indicating whether to print progress messages, including a progress bar that advances as each bin is processed. Default is TRUE.

Details

This function wraps the `extract_slim_features` workflow from the `ifcb-features` Python repository, which can be found at <https://github.com/WHOIGit/ifcb-features>.

Python and the `ifcb-features` package must be installed to use this function. The required Python packages can be installed in a virtual environment using `ifcb_py_install(features = TRUE)`, which additionally installs `ifcb-features` and its dependencies (a raw-data reader, `phasepack`, `scikit-image`, `scikit-learn`).

Supported ifcb-features versions: raw data is read through whichever reader the installed `ifcb-features` release provides - `ifcbkit` for v1.1.0 and later, `pyifcb` for v1.0.0 and earlier. Both are supported and may be installed side by side, with `ifcbkit` preferred when both are present. Use the `backend` argument (or the `IRFCB_IFCB_BACKEND` environment variable) to force a particular reader.

The feature code itself is unchanged between these releases, so the choice of reader does not affect how a region of interest is measured. The readers do not agree in every case, however: `pyifcb` skips a ROI whose recorded width is zero, while `ifcbkit` skips one whose width *or* height is zero, and `ifcbkit` additionally stitches overlapping ROI pairs in older I-style bins, which `pyifcb` returns separately. For the D-style bins produced by current instruments the two agree on ROI numbering and pixel data, and outputs are interchangeable; for I-style data, pin a reader with `backend` if you need results comparable to an earlier run.

Python version requirement: `ifcb-features` requires Python ≥ 3.10 . Installing v1.0.0 or earlier additionally pulls in `pyifcb`, which needs a binary `h5py` wheel (available for Python 3.10-3.13). See <https://github.com/WHOIGit/ifcb-features> for current requirements, and use `ifcb_py_install(features = TRUE)` to install into a compatible environment.

Multiblob output: the slim feature table describes each ROI's largest blob (plus summed* columns over all blobs). With `multiblob = TRUE`, the per-blob features of every blob in a multi-blob ROI are additionally written to `multiblob/<bin>_multiblob_v4.csv` inside `features_folder`, one row per blob with `roi_number`, `blob_number` and 18 morphological columns - the sidecar output `ifcb-features` introduced in v1.2.0, which is also the minimum version required (older releases never compute per-blob rows, and the function stops with an error if one is installed; update with `ifcb_py_install(features = TRUE)`). As upstream, a bin in which no ROI has more than one blob gets no sidecar file at all, so the presence of a `<bin>_multiblob_v4.csv` means that bin genuinely contains multi-blob ROIs. The skip logic accounts for this by reading the `numBlobs` column of a bin's existing feature CSV to tell whether a sidecar is expected: re-running with `multiblob = TRUE` over a directory previously extracted without it therefore skips the bins with single-blob ROIs only and re-extracts just those that need a sidecar, without `overwrite = TRUE`.

Bins are processed sequentially by default. When `parallel = TRUE`, bins are distributed across `n_cores` workers, which can substantially reduce runtime for large datasets. Existing outputs are skipped unless `overwrite = TRUE`, so the function can be re-run to resume an interrupted extraction.

The parallel backend depends on the platform. On Linux, bins run in separate worker processes, giving true multi-core parallelism. On Windows and macOS, where the embedded Python interpreter cannot reliably spawn worker processes, a thread pool is used instead; because of Python's Global Interpreter Lock the speedup there is smaller and depends on how much of the work runs in native (numpy / scikit-image) code. A further consequence of the thread backend is that interrupting a run (ESC / Stop) does not halt a bin already being processed: it finishes and writes its outputs before the run stops.

Value

Invisibly returns a tibble with one row per bin and the columns `bin`, `status` ("processed", "skipped" or "error") and `message`. The function is primarily called for its side effect of writing feature and blob files to disk.

See Also

`ifcb_py_install`, `ifcb_read_features`, <https://github.com/WHOIGit/ifcb-features>

Examples

```
## Not run:
# Install the Python environment including ifcb-features
ifcb_py_install(features = TRUE)

# Extract features and blobs from all bins in a data folder
ifcb_extract_features(
  data_folder = "path/to/data",
  features_folder = "path/to/features",
  blobs_folder = "path/to/blobs"
)

# Process a subset of bins in parallel using 4 cores
ifcb_extract_features(
  data_folder = "path/to/data",
  features_folder = "path/to/features",
```

```

blobs_folder = "path/to/blobs",
bins = c("D20220522T003051_IFCB134", "D20220522T000439_IFCB134"),
parallel = TRUE,
n_cores = 4
)

# Write IFCB Dashboard-compatible feature names (<bin>_fea_v4.csv)
ifcb_extract_features(
  data_folder = "path/to/data",
  features_folder = "path/to/features",
  blobs_folder = "path/to/blobs",
  feature_tag = "fea"
)

# Also write per-blob features for multi-blob ROIs
# (path/to/features/multiblob/<bin>_multiblob_v4.csv;
# requires ifcb-features >= 1.2.0)
ifcb_extract_features(
  data_folder = "path/to/data",
  features_folder = "path/to/features",
  blobs_folder = "path/to/blobs",
  multiblob = TRUE
)

## 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: "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)
```

<code>ifcb_get_mat_names</code>	<i>Get Variable Names from a MAT File</i>
---------------------------------	---

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 R reader, 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()`.

Otherwise, the function reads the .mat file with the default R reader.

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 R reader, 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().

Otherwise, the function reads the .mat file with the default R reader.

Value

A character vector of class names.

References

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

See Also

[ifcb_get_mat_names](#) <https://github.com/hsosik/ifcb-analysis>

Examples

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

# 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

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

Details

For a detailed example of a data submission, see [ifcb_get_shark_example](#).

Value

An empty data frame containing the SHARK column names.

See Also

[ifcb_get_shark_example](#)

Examples

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

`ifcb_get_shark_example`*Get Shark Column Example*

Description

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

ifcb_get_trophic_type(taxa_list)
```

<code>ifcb_is_diatom</code>	<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,
  details = FALSE,
  fuzzy = deprecated(),
  verbose = TRUE
)
```

Arguments

<code>taxa_list</code>	A character vector containing the list of taxa names.
<code>diatom_class</code>	A character string or vector specifying the class name(s) to be identified as diatoms, according to WoRMS. Default is "Bacillariophyceae".
<code>diatom_include</code>	Optional character vector of taxa (or genera) that should always be treated as diatoms, overriding the WoRMS-based classification. Default is NULL.
<code>max_retries</code>	An integer specifying the maximum number of attempts to retrieve WoRMS records in case of an error. Default is 3.
<code>sleep_time</code>	A numeric value indicating the number of seconds to wait between retry attempts. Default is 10 seconds.
<code>marine_only</code>	Logical. If TRUE, restricts the search to marine taxa only. Default is FALSE.
<code>details</code>	Logical. If TRUE, return a data frame with the resolved WoRMS class for each taxon instead of a logical vector. This is useful for auditing genus homonyms, i.e. diatom genera (such as <i>Navicula</i> or <i>Actinocyclus</i>) whose names are shared with animals and may therefore resolve to a non-diatom class in WoRMS. Inspect the <code>worms_class</code> column to spot such cases and add the affected taxa to <code>diatom_include</code> . Default is FALSE.
<code>fuzzy</code>	[Deprecated] The fuzzy argument is no longer available
<code>verbose</code>	A logical indicating whether to print progress messages. Default is TRUE.

Value

If `details = FALSE` (the default), a logical vector indicating whether each cleaned taxa name belongs to the specified diatom class. If `details = TRUE`, a [tibble](#) with one row per input taxon and the columns `taxa` (the original input), `genus` (the genus name used for matching), `worms_class` (the class resolved from WoRMS, or NA if no record was found) and `is_diatom` (the logical classification, including any `diatom_include` override).

See Also

<https://www.marinespecies.org/>

Examples

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

# Requires an internet connection to the WoRMS API; wrapped in try() so the
# example degrades gracefully when the service is unavailable.
try({
  res <- ifcb_is_diatom(taxa_list)
  print(res)
})
```

ifcb_is_in_basin	<i>Check if Points are in a Specific Sea Basin</i>
------------------	--

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.

<code>small_island_threshold</code>	Area threshold in square meters below which islands will be considered small and removed, if <code>remove_small_islands</code> is set to <code>TRUE</code> . Default is 2,000,000 (2 km ²).
<code>plot</code>	A logical indicating whether to plot the points, land polygon and buffer. Default is <code>FALSE</code> .
<code>verbose</code>	A logical indicating whether to print progress messages. Default is <code>TRUE</code> .
<code>utm_zone</code>	[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

[Deprecated]

The `api/list_bins` endpoint was removed from the upstream IFCB Dashboard ([WHOIGit/ifcddb@8c5839f1](https://github.com/WHOIGit/ifcddb@8c5839f1), 2026-03-08), so this function no longer works against the WHOI dashboard and other deployments tracking upstream. Use `ifcb_download_dashboard_metadata()` instead, which retrieves the same per-bin information from the still-supported `api/export_metadata` endpoint.

Usage

```
ifcb_list_dashboard_bins(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 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

```
## Not run:
# Deprecated: the upstream IFCB Dashboard removed `api/list_bins` on 2026-03-08.
bins <- ifcb_list_dashboard_bins("https://ifcb-data.who.edu/",
                                dataset_name = "mvco")

head(bins)

## End(Not run)
```

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")

# Requires an internet connection to the WoRMS API; wrapped in try() so the
# example degrades gracefully when the service is unavailable.
try({
  # 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.

<code>class2use_file_additions</code>	Character. Path to the <code>class2use</code> 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 <code>class2use_file_additions</code> that are not already included in the <code>class2use_file_base</code> will be added to generate the <code>class2use_file_output</code> .
<code>class2use_file_output</code>	Character. Path where the merged <code>class2use</code> file will be saved. If NULL, the merged file will be stored in the same directory as <code>class2use_file_base</code> . Default is NULL.
<code>manual_folder_base</code>	Character. Path to the folder containing the base set of manual classification .mat files.
<code>manual_folder_additions</code>	Character. Path to the folder containing the additions set of manual classification .mat files.
<code>manual_folder_output</code>	Character. Path to the output folder where the merged classification files will be stored.
<code>do_compression</code>	A logical value indicating whether to compress the .mat file. Defaults to TRUE.
<code>temp_index_offset</code>	Numeric. A large integer used to generate temporary indices during the merge process. Default is 50000.
<code>skip_class</code>	Character. A vector of class names to skip from the <code>class2use_file_additions</code> during the merge process. Default is NULL.
<code>quiet</code>	Logical. If TRUE, suppresses output messages. Default is FALSE.

Details

The .mat files are read, merged and written directly from R.

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

<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_HHMMSS" to the new format (YYYYMMDDTHHMMSS_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_HHMMSS") by inserting four empty columns after column 7 to match the newer format. [Experimental]
quiet	Logical. Suppress messages if TRUE (default: FALSE).

Details

The .mat classification files are written directly from R.

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_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 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/who_i_plankton/ifcb_merge_manual_ifcb_download_who_i_plankton_ifcb_download_dashboard_data

Examples

```
## Not run:
# Download and prepare WHOI-Plankton for the years 2013 and 2014
ifcb_prepare_who_i_plankton(
  years = c("2013", "2014"),
  png_folder = "who_i_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. The value is the index of the first 1 micron ESD bin included in the curve fit, calibrated for version 2 features. See the Feature file version section before using it with another version.
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. The function reads other versions unchanged, but the default start_fit and flag thresholds only apply to version 2. Re-derive them first; see the Feature file version section. [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.

Feature file version

The PSD is built from one feature, `EquivDiameter`, converted to microns with `micron_factor` and binned into 1 micron bins. The fitted curve, `max_ESD_diff` and every flag derived from them all follow from that single distribution.

`EquivDiameter` is not directly comparable between feature set versions. The version 4 extractor (`ifcb-features`, see `ifcb_extract_features()`) segments a tighter blob boundary than the version 2 MATLAB extractor (`ifcb-analysis`), so it reports a smaller equivalent diameter for the same image. The difference is roughly a fixed number of pixels rather than a fixed proportion. It therefore converts to microns through `micron_factor`, and moves the whole ESD histogram towards smaller bins by about that many microns. Small particles lose a larger share of their diameter than large ones.

The thresholds published by Hayashi et al. (2025) were calibrated on version 2 features and do not carry over unchanged:

- `start_fit` selects a histogram bin. When the distribution moves, the fit window sits at a different position relative to the peak, which changes the fitted exponent as well as the multiplier. The `r_sqr` and `beads` flags read those two values.
- `bloom` and `bubbles` are evaluated against `max_ESD_diff`, which is measured relative to `start_fit`. They move with the shifted distribution, and again with any compensating change to `start_fit`.
- `biomass` and `incomplete` depend on the height of the most populated bin, which changes as the shift moves targets between bins.

With a version other than 2, measure the offset on your own data instead of assuming a published value: extract both feature versions for a set of representative bins, compare `EquivDiameter` per ROI, and lower `start_fit` by the median difference in whole microns. Re-derive the flag

thresholds on that basis, and report the feature version together with any quality flags you publish. `micron_factor` sets what the offset amounts to in microns, so an instrument with a different pixel size needs its own value.

The file name is the only other place the version is used: features are read as `<bin>_fea_v<fea_v>.csv`. `ifcb_extract_features()` writes `<bin>_features_v4.csv` by default, so pass `feature_tag = "fea"` there to get the names this function searches for.

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`, `ifcb_extract_features`, <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	<i>Generate PSD Plot for a Given Sample</i>
---------------	---

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

sample_name	The name of the sample to plot in DYYYYMMDDTHHMMSS_IFCBXXX.
data	A data frame containing the PSD data (data output from ifcb_psd), where each row represents a sample and each column represents different particle sizes in micrometers.
fits	A data frame containing the fit parameters for the power curve (fits output from ifcb_psd), where each row represents a sample and the columns include the parameters a, k, and R^2.
start_fit	The x-value threshold below which data should be excluded from the plot and fit.
flags	Optional data frame or tibble with columns sample and flag. If sample_name appears in flags\$sample, the corresponding flag 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/kudelalab/PSD) <https://github.com/kudelalab/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,
    features = FALSE,
    features_ref = NULL
)
```

Arguments

envname	A character string specifying the name of the virtual environment to create. Default is "~/virtualenvs/iRfcb".
use_venv	Logical. If TRUE (default), a virtual environment is created. If FALSE, the system Python is used instead, and missing packages are installed globally for the user.
packages	A character vector of additional Python packages to install. If NULL (default), only the packages from "requirements.txt" are installed.
features	Logical. If TRUE, additionally installs the WHOI ifcb-features package (https://github.com/WHOIGit/ifcb-features) from GitHub, together with its dependencies (a raw-data reader, phasepack, scikit-image, scikit-learn). This is required by ifcb_extract_features(). Default is FALSE to keep the default environment lightweight. When installing into an existing virtual environment, the (slow) install is skipped if ifcb-features already imports successfully, unless features_ref is given.
features_ref	A character string specifying which git reference (release tag, branch, or commit) of ifcb-features to install when features = TRUE. If NULL (default), the latest published GitHub release is installed, which is more stable than the actively developed default branch. Use features_ref = "main" to install the latest development commit, or a tag such as "v1.0.0" to pin a specific version. The choice of reference determines which raw-data reader is installed: ifcb-features v1.1.0 and later depend on ifcbkit, while v1.0.0 and earlier depend on pyifcb. iRfcb supports both, so either reference works and the two readers may coexist in one environment. The feature code itself is unchanged between these releases, so the choice does not in itself affect how a region of interest is measured, and for the D-style bins produced by current instruments the two readers agree. They differ for older I-style bins, which ifcbkit stitches and pyifcb does not, and on ROIs with a zero height; see ifcb_extract_features(), whose backend argument pins a reader when both are installed. The reference does, however, decide which numpy version is installed, and that changes how a degenerate (collinear) blob is measured. Pinning "v1.0.0" pulls in pyifcb, whose dependencies resolve numpy to below 2.3, where numpy.linalg.eig() returns real eigenvalues and such a blob is reported as NaN in the Eccentricity, MajorAxisLength and MinorAxisLength columns. The ifcbkit-based releases leave numpy unconstrained, so a current install measures the same blob as 0, matching upstream ifcb-features. Pin "v1.0.0", or numpy < 2.3, to reproduce a run made before iRfcb 0.10.0; see ifcb_extract_features().

Note that installing v1.0.0 or earlier pulls in pyifcb, which requires binary wheels for h5py (available for Python 3.10-3.13). Installation fails on Python versions without such a wheel. The ifcbkit-based releases have no such constraint and require only Python ≥ 3.10 .

Whichever reference is used, scikit-image is constrained to < 0.28 , because ifcb-features calls morphology functions that are scheduled for removal in that release.

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 "TRUE" to automatically activate an installed Python venv when the iRfcb package is loaded. By default this activates a venv named iRfcb found in `reticulate::virtualenv_root()` (available via `reticulate::virtualenv_list()`; see examples). To activate a specific environment instead, also set the IRFCB_PYTHON_VENV variable to either the name of a venv under `reticulate::virtualenv_root()` or a full path to a venv directory. Both variables can be set in your .Renvirom file to enable automatic setup across sessions. 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"))

# Install the iRfcb Python venv including the WHOI ifcb-features package
# (latest release by default)
ifcb_py_install(envname = envpath, features = TRUE)

# Install a specific ifcb-features version, or the development branch
ifcb_py_install(envname = envpath, features = TRUE, features_ref = "v1.0.0")
ifcb_py_install(envname = envpath, features = TRUE, features_ref = "main")

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

## End(Not run)
```

ifcb_qc_sample	<i>Quality-control a raw IFCB sample (hdr/adc/roi triplet)</i>
----------------	--

Description

Validates the integrity and self-consistency of one or more raw IFCB samples and returns a tidy tibble of QC metrics and flags, one row per sample. Each sample is expected to consist of the standard IFCB file triplet sharing a base name: a header (.hdr), an ADC table (.adc), and the raw image data (.roi).

Usage

```
ifcb_qc_sample(
  sample,
  data_folder = NULL,
  max_ml = NULL,
  volume_tolerance = 0.05,
  runtime_tolerance = 0.02,
  max_roi_mb = NULL,
  max_humidity = NULL,
  max_temperature = NULL,
  flowrate = 0.25
)
```

Arguments

sample	Sample(s) to check. Either a single directory (all .adc files within are discovered recursively), or a character vector of sample base names or paths (with or without a .hdr/.adc/.roi extension). When data_folder is supplied, bare sample names are resolved against it.
data_folder	Optional directory in which to locate the triplet files when sample is given as bare sample names. Searched recursively.
max_ml	Optional fixed upper bound (in millilitres) for a plausible analyzed volume, applied to every sample. Default NULL derives the ceiling per sample from the header syringe volume (SyringeSampleVolume, falling back to syringeSize, then the 5 mL IFCB standard when neither is present or the reported volume is not positive) scaled by volume_tolerance. Set this only to override that instrument-reported value.
volume_tolerance	Fractional tolerance added to the derived syringe volume ceiling (default 0.05, i.e. 5%) to absorb estimation noise in the analyzed volume. Ignored when max_ml is supplied.
runtime_tolerance	Fractional slack by which the ADC's last-trigger run time may exceed the header's total run time before runtime_consistent is set to FALSE (default 0.02, i.e. 2%).

max_roi_mb	Optional numeric upper bound (in megabytes, where 1 MB = 1024 ² bytes) for the .roi file size. When supplied, samples whose .roi exceeds this size are flagged in the advisory roi_oversized column (e.g. max_roi_mb = 5 for 5 MB). Default NULL disables the check (roi_oversized is NA).
max_humidity	Optional numeric threshold (percent) for the header's recorded humidity. Samples above it are flagged in the advisory humidity_high column. Default NULL disables the check.
max_temperature	Optional numeric threshold (degrees, as recorded by the instrument) for the header's temperature. Samples above it are flagged in the advisory temperature_high column. Default NULL disables the check.
flowrate	Syringe flow rate (millilitres per minute) passed to <code>ifcb_volume_analyzed()</code> . Default 0.25.

Details

The checks build directly on the package's existing readers (`ifcb_read_hdr_data()`, `read_adc_columns()`, `ifcb_volume_analyzed()`) cover four areas:

Triplet completeness Whether all of .hdr, .adc, and .roi are present (`files_complete`).

ROI count consistency The number of imaged ROIs in the ADC (rows with a non-zero ROI width) must equal the header's roiCount (`roi_count_match`). Note that the ADC row count is **not** compared to triggerCount: depending on the ADC format a single trigger may yield several ROIs, so triggerCount is reported but not used as a hard check.

ROI data completeness The .roi file must be at least as large as the last image's end offset (`max(StartByte + width * height)`) computed from the ADC. A smaller file indicates a truncated or aborted transfer (`roi_data_complete`).

ROI dimension validity Every ROI width, height and start byte in the ADC must parse as a number (`roi_dims_valid`, with the offending row count in `n_roi_malformed`). A blank or NaN dimension is excluded from `n_rois` so that one damaged file cannot abort a whole survey, but the exclusion is reported rather than silent: without this check a width column that failed to parse would be indistinguishable from a sample that never triggered.

Run time consistency The run time recorded at the ADC's last trigger must not exceed the header's total run time (within `runtime_tolerance`), since a trigger cannot fire after acquisition has stopped (`runtime_consistent`). A header run time materially shorter than the ADC's last trigger points to corrupted or truncated header metadata; a run that legitimately continued past the last trigger (header run time longer than the ADC's) is normal for sparse samples and is not flagged.

Flow / volume sanity The analyzed volume from `ifcb_volume_analyzed()` must be positive and not exceed the syringe sample volume (`volume_ok`). The ceiling is taken from the header's `SyringeSampleVolume` (reported as `syringe_ml`, e.g. 5 mL for a standard IFCB), plus `volume_tolerance`, since the analyzed volume can never physically exceed the drawn syringe volume. A fixed ceiling can be forced with `max_ml`.

Further advisory flags are reported but do **not** affect `qc_pass`, as they describe valid samples that a user may nonetheless wish to exclude: `is_bead_run` (a bead/calibration run, detected from the header's `runBeads` field or a `sampleType` containing "bead"), `is_empty` (no imaged ROIs),

roi_oversized (the .roi file exceeds max_roi_mb, useful for catching overloaded or anomalous runs), and humidity_high / temperature_high (the header's recorded humidity / temperature exceed max_humidity / max_temperature, flagging possible condensation or overheating). The latter three are only evaluated when their threshold is supplied; otherwise they are NA. The measured humidity and temperature are always reported.

qc_pass is the conjunction of the integrity checks above (files_complete, roi_count_match, roi_data_complete, roi_dims_valid, runtime_consistent, volume_ok). A check that cannot be evaluated for a given sample is reported as NA and treated as *not applicable*: it does not fail qc_pass. This matters for legacy IFCB headers, which omit the post-run roiCount summary field (so roi_count_match is NA); such samples can still pass on the checks that do apply. It also applies to a sample that never triggered, where no volume can be computed and volume_ok is NA rather than FALSE (is_empty reports the condition instead). A sample whose analyzed volume comes out as exactly 0 is a different case and does fail: that is a computed answer saying the instrument ran and analyzed no water, rather than a check that could not be run. Only a check that actually evaluates to FALSE fails the sample. files_complete is always TRUE/FALSE (never NA) and so always counts.

Value

A tibble with one row per sample containing the resolved file paths, QC metrics, boolean QC flags, and an overall qc_pass column.

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_read_hdr_data()` `ifcb_volume_analyzed()`

Examples

```
## Not run:
# Check every sample in a data directory
qc <- ifcb_qc_sample("data/raw")

# Keep only clean, non-bead samples for analysis
dplyr::filter(qc, qc_pass, !is_bead_run)

## 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

<code>feature_files</code>	A path to a folder containing feature files or a character vector of file paths.
<code>multiblob</code>	Logical indicating whether to filter for multiblob files (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>biovolume_only</code>	Logical; if TRUE, only a minimal set of feature columns required for biovolume calculations are read from each feature file (typically <code>roi_number</code> and <code>biovolume</code>). This substantially reduces memory usage and improves performance when other features are not needed. If FALSE, all available feature columns are read.
<code>verbose</code>	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 <code>hdr_files</code> instead, which also accepts a single folder path. This rename applies to this function only, because <code>hdr_files</code> takes a vector of file paths as well as a folder. It is not a package-wide rename: functions that genuinely take one directory, such as <code>ifcb_psd()</code> and <code>ifcb_summarize_biovolumes()</code> , keep <code>hdr_folder</code> .

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*Read a MATLAB .mat File in R*

Description

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

Usage

```
ifcb_read_mat(file_path)
```

Arguments

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

Details

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)
```

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

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

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 R reader, 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()`.

Otherwise, the function reads the .mat file with the default R reader.

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

The MAT files are read and written directly from R, producing output identical to the MATLAB ifcb-analysis format.

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

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

Examples

```
## Not run:  
# 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://ifcb.serve.scilifelab.se",
  model_name = "SMHI NIVA SYKE SAMS SZN ResNet 50 V6",
  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://ifcb.serve.scilifelab.se", an instance hosted on the SciLife-Lab Serve platform. A free example Hugging Face Space is also available at

	"https://irfcb-classify.hf.space" (limited resources, intended for testing and demonstration). For large-scale or production 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 SYKE SAMS SZN ResNet 50 V6". Use ifcb_classify_models() 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).

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. Written directly from R.
- "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,
  diatom_equation = c("large", "all", "auto"),
  threshold = "opt",
  feature_recursive = TRUE,
  class_recursive = TRUE,
  hdr_recursive = TRUE,
  drop_zero_volume = FALSE,
  feature_version = NULL,
  use_cell_counts = FALSE,
  single_cell_values = c(-1, 0),
  carbon_conversion = c("roi", "cell"),
  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. Supply a single file format per sample: a sample represented twice (e.g. by both a .mat and a .h5) would have its ROIs counted once per file, so this is rejected with an error naming the affected samples.
<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.
<code>hdr_folder</code>	(Optional) Path to the folder containing HDR files. Needed for calculating cell, biovolume and carbon concentration per liter.
<code>custom_images</code>	(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>micron_factor</code>	Conversion factor from microns per pixel (default: 1/3.4).
<code>diatom_class</code>	A character vector of diatom class names in the World Register of Marine Species (WoRMS). Default is "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 is FALSE.
<code>diatom_equation</code>	A character string selecting which Menden-Deuer and Lessard (2000) carbon-to-volume relationship to apply to diatoms. "large" (default) uses the large-diatom ($> 3000 \text{ micron}^3$) equation, matching the <code>ifcb-analysis</code> convention. "all" uses the all-sizes diatom equation, which assigns more carbon to small cells. "auto" selects between them per volume (large-diatom above 3000 micron^3 , all-sizes at or below), keeping each in its calibrated range at the cost of a discontinuity at the boundary, where predicted carbon falls from about 190 to 135 pgC as volume increases. Note that biovolume is measured per region of interest (ROI/image), not per cell, so chains of small cells register a large ROI biovolume unless <code>carbon_conversion = "cell"</code> is used. Passed to <code>ifcb_extract_biovolumes</code> .
<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>feature_recursive</code>	Logical. If TRUE, the function will search for feature files recursively within the <code>feature_folder</code> . 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_cell_counts	Logical. If TRUE, reads the optional per-ROI cell_count data stored by the diatom chain counter in .mat/.h5/.csv classification files and adds cell_counts (and cell_counts_per_liter when hdr_folder is supplied) to the output, reporting cell abundance alongside ROI counts. Only supported with automated class_files. The function aborts if enabled but no classification file contains chain-count data. For chain-length statistics (mean, median, max chain length) use ifcb_summarize_cell_counts . Note that cell_counts here is summed only over ROIs that also have matching feature (biovolume) data (the same ROI population as counts); ifcb_summarize_cell_counts instead sums over all classified ROIs, so the two abundance totals can differ. Default is FALSE.
single_cell_values	Integer vector of cell_count values that should be treated as a single cell when computing cell_counts. Default is c(-1, 0), i.e. ROIs that were not counted (-1) and ROIs where no cells were detected (0) each count as one cell. Values not listed are used verbatim. Only used when use_cell_counts = TRUE.
carbon_conversion	A character string controlling how the Menden-Deuer and Lessard (2000) relationships are applied. "roi" (default) applies the selected equation once to the whole ROI biovolume, matching the ifcb-analysis convention and reproducing previous results exactly. "cell" applies it to the per-cell volume and sums over the chain, which is how the relationships are defined; it requires use_cell_counts = TRUE. carbon_ug remains a per-class total in both cases. Unlike cell_counts, it is not NA for a sample whose classification file carries no cell_count data: those ROIs are converted as single cells, which is the value reported today rather than an unknown. See ifcb_extract_biovolumes for the full rationale.
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. The diatom relationship is selected with `diatom_equation` ("large", the default, or "all"). 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 R reader, 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'. When `use_cell_counts = TRUE`, the cell abundance columns 'cell_counts' (and 'cell_counts_per_liter' when `hdr_folder` is provided) are also included. `cell_counts` is NA for a sample whose classification file carries no `cell_count` data, since the cell total is unknown there; it is not reported as 0, which would be indistinguishable from a taxon that was genuinely absent.

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.
- Groves, G. J. J., Arthur, G., Bresnan, E., Whyte, C., Arce, P. and Davidson, K. (2026), Automatic enumeration of chains of marine diatoms using "You Only Look Once" - a machine learning approach. *Journal of Plankton Research*, 48(2), fbaf064, doi: 10.1093/plankt/fbaf064.

See Also

`ifcb_summarize_cell_counts` `ifcb_extract_biovolumes` <https://github.com/nodc-sweden/ifcb-pytorch-classify>

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_cell_counts

Summarize Diatom Cell Counts and Chain-Length Statistics from IFCB Data

Description

Summarizes the optional per-ROI cell-count data produced by the diatom chain counter and stored in classification files (.mat, .h5 or .csv). For each sample and class it computes the total cell abundance (number of cells, accounting for chains) together with a user-selectable set of chain-length statistics.

Usage

```
ifcb_summarize_cell_counts(
  class_files,
  hdr_folder = NULL,
  single_cell_values = c(-1, 0),
  stats = c("n_counted", "mean", "median", "max"),
  threshold = "opt",
  class_recursive = TRUE,
  hdr_recursive = TRUE,
  use_python = FALSE,
  verbose = TRUE
)
```

Arguments

<code>class_files</code>	A character vector of full paths to classification files (<code>.mat</code> , <code>.h5</code> or <code>.csv</code>), or a single path to a folder containing such files. Any of these file types can carry the optional <code>cell_count</code> data written by the chain counter; files without it are treated as NA chain counts. Supply a single file format per sample: a sample represented twice (e.g. by both a <code>.mat</code> and a <code>.h5</code>) would have its ROIs counted once per file, so this is rejected with an error naming the affected samples.
<code>hdr_folder</code>	(Optional) Path to the folder containing HDR files. Needed for calculating cell abundance per liter.
<code>single_cell_values</code>	Integer vector of <code>cell_count</code> values that should be treated as a single cell when computing abundance. Default is <code>c(-1, 0)</code> , i.e. both ROIs that were not counted and ROIs where no cells were detected count as one cell. Values not listed are used verbatim.
<code>stats</code>	Character vector selecting which chain-length statistics to include. Any of "n_counted" (the number of ROIs the chain counter measured, which the other statistics are computed over), "mean", "median", "max", and "sd". Default is <code>c("n_counted", "mean", "median", "max")</code> . Use <code>character(0)</code> to return abundance only.
<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.
<code>class_recursive</code>	Logical. If TRUE and <code>class_files</code> is a folder, searches recursively for classification files. Default is TRUE.
<code>hdr_recursive</code>	Logical. If TRUE, searches for HDR files recursively within <code>hdr_folder</code> (if provided). Default is TRUE.
<code>use_python</code>	Logical. If TRUE, attempts to read <code>.mat</code> files using a Python-based method (SciPy). Default is FALSE.
<code>verbose</code>	Logical. If TRUE, prints progress messages. Default is TRUE.

Details

The chain counter stores one integer `cell_count` per region of interest (ROI). The value `-1` marks ROIs of classes that were not configured for chain counting, `0` marks ROIs that were counted but where no cells were detected, and a positive value is the number of cells in that ROI. Abundance is derived by translating the values listed in `single_cell_values` to a single cell and using every other value verbatim (see `ifcb_summarize_biovolumes()`, which shares this logic to report `cell_counts`).

Chain-length statistics (mean, median, max, sd) are computed only over ROIs that were genuinely chain-counted (`cell_count >= 1`); ROIs with `-1` (not counted) or `0` (no cells detected) are excluded from the length statistics, although both still contribute to abundance according to `single_cell_values` (by default one cell each).

`n_counted` reports how many ROIs those length statistics were computed over, i.e. the number of ROIs the chain counter measured (`cell_count >= 1`). It is a count of ROIs rather than of chains, and

it includes ROIs found to hold a single cell, which are not chains. It is useful for telling a measured abundance from an imputed one: a class with `cell_counts > 0` but `n_counted == 0` was never chain-counted, so its abundance is one cell per ROI by imputation rather than by measurement.

Chain counting was introduced by Groves et al. (2026), who trained a "You Only Look Once" (YOLO) object detection model to enumerate the cells in diatom chains imaged by the IFCB. The per-ROI `cell_count` data summarized here is produced by the `ifcb-pytorch-classify` inference pipeline (<https://github.com/nodc-sweden/ifcb-pytorch-classify>), which writes it as an optional `cell_count` variable in the `.mat`, `.h5` and `.csv` classification files alongside the class predictions.

This function derives `cell_counts` from every classified ROI. This differs from `ifcb_summarize_biovolumes()`, which reports `cell_counts` only over ROIs that also have matching feature (biovolume) data, so the two abundance totals can differ when some ROIs lack feature data.

Value

A data frame with one row per sample and class. Columns always include `sample`, `classifier`, `class`, `counts` (number of ROIs), and `cell_counts` (total cell abundance). The requested chain-length statistics are added as `n_counted` (number of ROIs the chain counter measured, i.e. those with `cell_count >= 1`, including single-cell ones), `mean_chain_length`, `median_chain_length`, `max_chain_length`, and/or `sd_chain_length`. When `hdr_folder` is provided, `ml_analyzed` and `cell_counts_per_liter` are also returned.

`cell_counts` is NA for a sample whose classification file carries no `cell_count` data, since the cell total is unknown there. It is not reported as 0, which would be indistinguishable from a taxon that was genuinely absent. `counts` is unaffected and still reports the ROIs.

References

Groves, G. J. J., Arthur, G., Bresnan, E., Whyte, C., Arce, P. and Davidson, K. (2026), Automatic enumeration of chains of marine diatoms using "You Only Look Once" - a machine learning approach. *Journal of Plankton Research*, 48(2), fbaf064, doi: 10.1093/plankt/fbaf064.

See Also

`ifcb_summarize_biovolumes` `ifcb_extract_biovolumes` <https://github.com/nodc-sweden/ifcb-pytorch-classify>

Examples

```
## Not run:
# Summarize chain counts and abundance from classification files
chains <- ifcb_summarize_cell_counts("path/to/class")

# Include abundance per liter and only the mean chain length
chains <- ifcb_summarize_cell_counts(
  "path/to/class",
  hdr_folder = "path/to/hdr",
  stats = "mean"
)
```

```
## 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 R reader, 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()`.

Otherwise, the function reads the .mat file with the default R reader.

Value

A list containing the following elements:

<code>class2useTB</code>	Classes used in the TreeBagger classifier.
<code>classcountTB</code>	Counts of each class considering each target placed in the winning class.

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

Examples

```
## Not run:
ifcb_summarize_class_counts('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.
------------	--

<code>hdr_folder</code>	A character string specifying the path to the directory containing the .hdr files. Default is NULL.
<code>sum_level</code>	A character string specifying the level of summarization. Options: "sample" (default) or "class".
<code>verbose</code>	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

<code>png_folder</code>	Character. The file path to the folder containing the PNG images.
<code>feature_folder</code>	Character. The file path to the folder containing the feature files (optional).
<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>hdr_folder</code>	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.

A value is NA when it cannot be derived from the file, for example when the ADC lacks the run/inhibit time columns (with a warning) or records no nonzero inhibit time, matching the NaN the MATLAB reference returns in those cases.

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	<i>Determine if Points are in a Specified Sea Basin</i>
------------------	---

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

image_folder	The directory containing subdirectories with image files.
output_dir	The directory where the zip archives will be written.
n_images	Integer. Maximum number of images to randomly sample per subdirectory. If NULL, all images are included.
quiet	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	<i>Zip PNG Folders</i>
---------------	------------------------

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	<i>Process IFCB String</i>
---------------------	----------------------------

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	<i>Function to Read Individual Files and Extract Relevant Lines</i>
---------------	---

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	<i>Summarize TreeBagger Classifier Results</i>
-------------------	--

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_diatom	<i>Convert Biovolume to Carbon for Diatoms (All Sizes)</i>
--------------	--

Description

This function converts biovolume in microns³ to carbon in picograms for diatoms across the full size range 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 $\log a = -0.541$ and $b = 0.811$.

Usage

```
vol2C_diatom(volume)
```

Arguments

volume A numeric vector of biovolume measurements in microns³.

Details

This relationship is fit to diatoms of all sizes and assigns a higher carbon density than [vol2C_lgdiatom](#) (which is specific to large diatoms larger than 3000 microns³). Because the large-diatom equation is intended only for cells > 3000 microns³, switching equations at that threshold introduces a discontinuity: at 3000 microns³ the all-sizes equation predicts ~190 pgC versus ~135 pgC for the large-diatom equation. (The two curves themselves only intersect near 4e5 microns³.)

Value

A numeric vector of carbon measurements in picograms.

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.

See Also

[vol2C_lgdiatom](#) for the large-diatom (> 3000 microns³) relationship.

Examples

```
# Volumes in microns^3
volume <- c(500, 1000, 2000)

# Convert biovolume to carbon for diatoms (all sizes)
vol2C_diatom(volume)
```

vol2C_diatom_auto	<i>Convert Biovolume to Carbon for Diatoms, Choosing the Equation by Volume</i>
-------------------	---

Description

This function converts biovolume in microns³ to carbon in picograms for diatoms, selecting between the two Menden-Deuer and Lessard (2000) diatom relationships element-wise: volumes greater than 3000 microns³ use the large-diatom equation ([vol2C_lgdiatom](#)), and the rest use the all-sizes equation ([vol2C_diatom](#)). It backs diatom_equation = "auto" in [ifcb_extract_biovolumes\(\)](#) and [ifcb_summarize_biovolumes\(\)](#), and applies to whatever volume it is handed: the ROI biovolume under carbon_conversion = "roi", or the per-cell volume under carbon_conversion = "cell".

Usage

```
vol2C_diatom_auto(volume)
```

Arguments

volume A numeric vector of biovolumes in microns³.

Details

Be aware that the two relationships are not continuous at the 3000 micron³ boundary: the all-sizes equation predicts about 190 pgC there and the large-diatom equation about 135 pgC. Selecting between them by volume therefore makes predicted carbon *drop* by roughly 41% as a cell grows across the boundary, which is why this is not the default. Use it when keeping each equation inside its calibrated size range matters more than a monotonic carbon-to-volume curve.

Value

A numeric vector of carbon content in picograms.

References

Menden-Deuer Susanne, Lessard Evelyn J., (2000), Carbon to volume relationships for dinoflagellates, diatoms, and other protist plankton, Limnology and Oceanography, 45(3), 569-579, doi: 10.4319/lo.2000.45.3.0569.

See Also

[vol2C_diatom](#) [vol2C_lgdiatom](#)

Examples

```
# Volumes in microns^3, spanning the 3000 micron^3 boundary
volume <- c(500, 2000, 5000, 20000)

# Small volumes use the all-sizes equation, large ones the large-diatom equation
vol2C_diatom_auto(volume)
```

vol2C_lgdiatom

Convert Biovolume to Carbon for Large Diatoms

Description

This function converts biovolume in microns³ to carbon in picograms for 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 $\log a = -0.933$ 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.

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.

See Also

[vol2C_diatom](#) for the all-sizes diatom relationship.

Examples

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

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

vol2C_nondiatom	<i>Convert Biovolume to Carbon for Non-Diatom Protists</i>
-----------------	--

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, 4

ifcb_adjust_classes, 4, 19
ifcb_annotate_batch, 5
ifcb_annotate_samples, 8
ifcb_annotate_samples(), 6
ifcb_classify_images, 10
ifcb_classify_images(), 11–14, 80
ifcb_classify_models, 11
ifcb_classify_models(), 10, 11, 13, 14, 80
ifcb_classify_sample, 12
ifcb_classify_sample(), 10–12, 80
ifcb_convert_filenames, 14, 90
ifcb_correct_annotation, 7, 15
ifcb_count_mat_annotations, 17
ifcb_create_class2use, 5, 9, 18
ifcb_create_manifest, 19
ifcb_create_manual_file, 7, 20, 61
ifcb_download_dashboard_data, 22, 61
ifcb_download_dashboard_data(), 24, 55
ifcb_download_dashboard_metadata, 24
ifcb_download_dashboard_metadata(), 23, 55
ifcb_download_test_data, 25
ifcb_download_who_i_plankton, 26, 60, 61
ifcb_extract_annotated_images, 27, 36, 42
ifcb_extract_biovolumes, 30, 83, 84, 87
ifcb_extract_biovolumes(), 103
ifcb_extract_classified_images, 29, 34, 42
ifcb_extract_features, 37, 65
ifcb_extract_features(), 68
ifcb_extract_pngs, 29, 36, 40
ifcb_extract_pngs(), 11–14, 29, 36, 80
ifcb_get_ecotaxa_example, 42
ifcb_get_ferrybox_data, 43
ifcb_get_mat_names, 44, 46
ifcb_get_mat_variable, 45, 46
ifcb_get_runtime, 47
ifcb_get_shark_colnames, 48, 49
ifcb_get_shark_example, 48, 49
ifcb_get_trophic_type, 49
ifcb_is_diatom, 34, 50
ifcb_is_in_basin, 52
ifcb_is_near_land, 53
ifcb_list_dashboard_bins, 55
ifcb_list_dashboard_bins(), 23, 24
ifcb_match_taxa_names, 56
ifcb_merge_manual, 57, 61
ifcb_prepare_who_i_plankton, 59
ifcb_psd, 62, 66
ifcb_psd(), 74
ifcb_psd_plot, 66
ifcb_py_install, 39, 65, 67, 75
ifcb_qc_sample, 70
ifcb_read_features, 34, 39, 72
ifcb_read_hdr_data, 74, 90
ifcb_read_hdr_data(), 71, 72
ifcb_read_mat, 75
ifcb_read_summary, 76
ifcb_replace_mat_values, 77
ifcb_run_image_gallery, 78
ifcb_save_classification, 79
ifcb_summarize_biovolumes, 81, 87
ifcb_summarize_biovolumes(), 74, 86, 87, 103
ifcb_summarize_cell_counts, 34, 83, 84, 85
ifcb_summarize_class_counts, 88
ifcb_summarize_png_counts, 89
ifcb_summarize_png_metadata, 91
ifcb_volume_analyzed, 92
ifcb_volume_analyzed(), 71, 72
ifcb_volume_analyzed_from_adc, 93
ifcb_which_basin, 94
ifcb_zip_images_by_class, 95
ifcb_zip_matlab, 96, 99
ifcb_zip_pngs, 98, 98

`process_ifcb_string`, [100](#)

`read_hdr_file`, [100](#)

`split_large_zip`, [101](#)

`summarize_TBclass`, [102](#)

`tibble`, [51](#)

`vol2C_diatom`, [102](#), [103–105](#)

`vol2C_diatom_auto`, [103](#)

`vol2C_lgdiatom`, [103](#), [104](#), [104](#)

`vol2C_nondiatom`, [105](#)