Package: LEEF.analysis (via r-universe)

October 10, 2024

Type Package

Title Access Functions, Tests and Basic Analysis of the RRD Data from the LEEF Project

Version 1.0.2

Maintainer Rainer M. Krug <`Rainer.Krug@uzh.ch`>

BugReports `https://github.com/LEEF-UZH/LEEF.analysis/issues`

URL `https://github.com/LEEF-UZH/LEEF.analysis`

Description Provides simple access functions to read data out of the sqlite RRD database. SQL queries can be configured in a yaml config file and used.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports DBI, RSQLite, yaml, parallel, ggplot2, rmarkdown, utils, dplyr, dbplyr, magrittr, rlang, tibble, pbapply, pbmcapply, quarto, data.table, stats, tidyr, parquetize, arrow

RoxygenNote 7.2.3

Suggests knitr, putrrr

VignetteBuilder knitr

Repository https://leef-uzh.r-universe.dev

RemoteUrl https://github.com/LEEF-UZH/LEEF.analysis

RemoteRef master

RemoteSha ddef64e85fb7d65223a3661ea0496de742acc7bc

Contents

`add_experiment_tables` *Add* `composition` *and* `experimetal_design` *table to db.*

Description

Add `composition` and `experimetal_design` table to db.

Usage

```
add_experiment_tables(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  composition = NULL,
  experimetal_design = NULL,
  light_decline_schedule = NULL,
  immigration_schedule = NULL,
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`composition`	fully qualified path to the `csv` file with the composition data
`experimetal_design`	fully qualified path to the `csv` file with the experimental design table
`light_decline_schedule`	fully qualified path to the `csv` file with the light decline dates
`immigration_schedule`	fully qualified path to the `csv` file with the immigration event dates
`overwrite`	if `TRUE`, the existing tables will be overwritten. There is actually no need for this.

`add_extracted_csv_to_db`

Title

Description

Title

Usage

```
add_extracted_csv_to_db(
  db,
  csv_file = "Morph_mvt_non_cropped.csv",
  measure = "bemovi.mag.25",
  tn_postfix = "non_cropped",
  archive_dir = "/Users/rainerkrug/Duck/LEEFSwift3",
  copy_locally_first = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database.Must be set!
`csv_file`	name of the csv files to be imported
`measure`	measure
`tn_postfix`	postfix for the table name,
`archive_dir`	directory containes the archive. It contains the following directories: - LEEF.archived.data - LEEF.archived.data_segments - LEEF.backend.data - LEEF.backend.data_segments
`copy_locally_first`	

`add_reclassified_to_db`

Title

Description

Title

Usage

```
add_reclassified_to_db(
  path,
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  remove_timestamps = NULL,
  check_timestamps = TRUE,
  backup_removed = TRUE,
  methods = NULL,
  DBIDriver = RSQLite::SQLite()
)
```

Arguments

`path`	path to the root directory of the reclassification
`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`remove_timestamps`	vector of timestamps to be removed.
`check_timestamps`	If ‘TRUE‘, the data will ony be added when timestamp does not exist in db yet. If ‘FALSE‘, it will always be added. Usually this should **NOT** be done.#’
`backup_removed`	if `TRUE` data which will be replaced will be backed up.
`DBIDriver`	the DBI driver to use. Default is RSQLite::SQLite()
`method`	method to be added. If `NULL`, method will be determined by the sub-directories.

`add_to_db` *Add data to RRD database into existing table*

Description

Add data to RRD database into existing table

Usage

```
add_to_db(
  fns,
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  tables,
  remove_timestamps = NULL,
  check_timestamps = TRUE,
  backup_removed = TRUE,
  DBIDriver = RSQLite::SQLite()
)
```

Arguments

`fns`	vector of `rds` or `csv` files names containing data to be added. The data has to contain a colum named `timestamp`.
`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`tables`	`vector` of the table names the data of the `fns` of the same index should be added to. **The table has to exist and contain a field named** `timestamp`
`remove_timestamps`	vector of timestamps to be removed.
`backup_removed`	if 'TRUE‘ data which will be replaced will be backed up.
`DBIDriver`	the DBI driver to use. Default is RSQLite::SQLite()
`check_timestamps.`	If `TRUE`, the data will ony be added when timestamp does not exist in db yet. If `FALSE`, it will always be added. Usually this should **NOT** be done.

Value

vector of length of `fns` with `TRUE` if the data has been added, `FALSE` otherwise

---

`arrow_read_conductivity` *Read* `conductivity` *table from an arrow database and return as a tibble.*

Description

This function returns the actual data.

Usage

```
arrow_read_conductivity(
  db = getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF-2/parquet/"),
  from_timestamp = 20221106,
  to_timestamp = 20230812
)
```

Arguments

`db`	fully qualified path to the folder which contains all the arrow directories, in case of LEEF called `parquet`. Defaul: `getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`

Value

`tibble` containing the data

`arrow_read_density` *Read* `density` *table from an arrow database and return as a tibble.*

Description

This function returns the actual data.

Usage

```
arrow_read_density(
  db = getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF-2/parquet/"),
  from_timestamp = 20221106,
  to_timestamp = 20230812
)
```

Arguments

`db`	fully qualified path to the folder which contains all the arrow directories, in case of LEEF called `parquet`. Defaul: `getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`

Value

`tibble` containing the data

`arrow_read_o2` *Read* `o2` *table from an arrow database and return as a tibble.*

Description

This function returns the actual data.

Usage

```
arrow_read_o2(
  db = getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF-2/parquet/"),
  from_timestamp = 20221106,
  to_timestamp = 20230812
)
```

Arguments

`db`	fully qualified path to the folder which contains all the arrow directories, in case of LEEF called `parquet`. Defaul: `getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`

Value

`tibble` containing the data

`arrow_read_table` *Read a table from an arrow database and return as an Dataset object.*

Description

This function does not actually fetch the data, but returns an Dataset object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`. This function is simply a convenience and consistency wrapper around `open_dataset`

Usage

```
arrow_read_table(
  table,
  db = getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF-2/parquet/"),
  hive_style = TRUE,
  unify_schemas = FALSE,
  format = c("parquet")
)
```

Arguments

`table`	Directory of the data files
`db`	fully qualified path to the folder which contains all the arrow directories, Defaul: `getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF-2/parquet/")`
`hive_style`	see `open_dataset`
`unify_schemas`	see `open_dataset`
`format`	see `open_dataset`

Value

A Dataset R6 object. Use dplyr methods on it to query the data.

See Also

`open_dataset`

`arrow_read_toc` *Read* `toc` *table from an arrow database and return as a tibble.*

Description

This function returns the actual data.

Usage

```
arrow_read_toc(
  db = getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF-2/parquet/"),
  from_timestamp = 20221106,
  to_timestamp = 20230812,
  duplicates = mean
)
```

Arguments

`db`	fully qualified path to the folder which contains all the arrow directories, in case of LEEF called `parquet`. Defaul: `getOption("RRDarrow", "/Volumes/RRD.Reclassification_LEEF`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`
`duplicates`	function which will be used to combine duplicates. If `NULL`, co summarising is done. Default is ‘mean‘.

Value

`tibble` containing the data

`CalculateDensities` *Re-calculate densities*

Description

Re-calculate densities

Usage

```
CalculateDensities(morph, meas, extrapolation.factor, cropping.factor)
```

Arguments

`meas`

`classify_bemovi_archive`

Classify bemovi data by using the archived data

Description

Classify bemovi data by using the archived data

Usage

```
classify_bemovi_archive(
  archive_dir = "/Users/rainerkrug/MountainDuck/LEEFSwift3",
  magnification = 16,
  bemovi_extract_name = "bemovi_extract.yml",
  species_tracked = NULL,
  timestamps,
  classifier_constant_name,
  classifier_increasing_name,
  output,
  exclude_videos = NULL,
  mc.cores = 1
)
```

Arguments

`magnification`	the magnification or the bemovi videos to be reclassified
`bemovi_extract_name`	the name of the `.yml` containing the parameter for the analysis in the directory of the data. The directory depends on the `root_dir`, `magnification` and `timestamp`
`species_tracked`	names of the species tracked as a character vector. If `NULL` the species tracked from the `bemovi_extract_file` will be used.
`timestamps`	`character` vector containing the timestamps to be classified
`classifier_constant_name`	`character` vector of name of the classifier for temperature treatment **constant** including path
`classifier_increasing_name`	`character` vector of name of the classifier for temperature treatment **increasing** including path
`output`	path to which the classified data will be saved as `rds`
`exclude_videos`	file names of videos to exclude. If `NULL`, all will be used.
`mc.cores`	number of cores to be used. Defaults to 1

Value

invisible `NULL`

`classify_bemovi_files` *Classify algae_traits data.frame*

Description

Classify algae_traits data.frame

Usage

```
classify_bemovi_files(
  datadir,
  bemovi_extract_name = NULL,
  classifier_constant_name,
  classifier_increasing_name,
  exclude_videos = NULL
)
```

Arguments

`datadir`	`character` vector containing the root directory for all files
`bemovi_extract_name`	`character` vector containing the name of the bemovi config file including path.
`classifier_constant_name`	`character` vector of name of the classifier for temperature treatment **constant** including path
`classifier_increasing_name`	`character` vector of name of the classifier for temperature treatment **increasing** including path
`exclude_videos`	file names of videos to exclude. If `NULL`, all will be used.

Value

`list` containing three objects: - `mean_density_per_ml` - `morph_mvt` - `trajectory_data`

`classify_flowcam_archive` *Classify algae_traits data.frame*

Description

Classify algae_traits data.frame

Usage

```
classify_flowcam_archive(
  archive_dir = "/Users/rainerkrug/MountainDuck/LEEFSwift3",
  timestamps,
  algae_traits_name = "algae_traits_filtered.rds",
  classifier_constant_name,
  classifier_increasing_name,
  species_tracked = NULL,
  output,
  mc.cores = 1,
  bottle = NULL
)
```

Arguments

`timestamps`	`character` vector containing the timestamps to be classified
`classifier_constant_name`	the classifier for temperature treatment **constant**
`classifier_increasing_name`	the classifier for temperature treatment **increasing**
`species_tracked`	names of the species tracked as a character vector. If `NULL` it will be read from the original configuration file in the `datadir`.
`output`	path to which the classified data will be saved as `rds`
`mc.cores`	number of cores to be used. Defaults to 1
`bottle`	if not 'NULL' (default) only classify this bottle. Needs to be a single bottle!

Value

invisible `NULL`

`classify_flowcam_files` *Classify algae_traits data.frame*

Description

Classify algae_traits data.frame

Usage

```
classify_flowcam_files(
  datadir,
  algae_traits_name = "algae_traits_filtered.rds",
  classifier_constant_name,
  classifier_increasing_name,
  timestamp = "55555555",
  species_tracked = NULL,
  bottle = NULL
)
```

Arguments

`datadir`	`character` vector containing the root directory for all files
`algae_traits_name`	`character` vector containing the name of the algae traits file without path
`classifier_constant_name`	`character` vector of name of the classifier for temperature treatment **constant** including path
`classifier_increasing_name`	`character` vector of name of the classifier for temperature treatment **increasing** including path
`timestamp`	timestamp to be used to stamp the classified data
`species_tracked`	names of the species tracked as a character vector. If `NULL` it will be read from the original configuration file in the `datadir`.
`bottle`	if not 'NULL' (default) only classify this bottle. Needs to be a single bottle!

Value

`list` containing two objects: - `algae_traits` including species - `algae_densities` densities of the different particles identified

---

`configs` *Return names of all configs in config file*

Description

Return names of all configs in config file

Usage

```
configs(config_yml = system.file("tables.yml", package = "LEEF.analysis"))
```

Arguments

`config_yml`	the config file containing the sql queries

Value

`ggplot` object of the plot

`create_rrd_trimmed` *Create trimmed down density-only version of the database*

Description

This function creates a trimmed down version of the database which does not contain the tables only necessary for the classification. It does not do anything with the original database.

Usage

```
create_rrd_trimmed(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  trimmed_db = gsub("\\.sqlite", ".trimmed.sqlite", db)
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`trimmed_db`	fully qualified path to the trimmed sqlite database. Defaults to the db with the extension `trimmed.sqlite` instead of `sqlite`.

Value

the path and name of the trimmed database

`db_disconnect` *Disconnect the* `tbl_SQLiteConnection`

Description

Disconnect the object returned by the `db_read_...` functions.

Usage

```
db_disconnect(x)
```

Arguments

`x`	a `tbl_SQLiteConnection` object

Value

invisibly `NULL`

`db_read_conductivity` *Read* `condi=uctivity` *from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_conductivity(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  from_timestamp = 20210920,
  to_timestamp = 21000101
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`

Value

Link to the table `density` in the RRD to be used with `dplyr` and friends

---

`db_read_density` *Read* `density` *from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_density(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  from_timestamp = 20210920,
  to_timestamp = 21000101
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`

Value

Link to the table `density` in the RRD to be used with `dplyr` and friends

---

`db_read_immigration` *Read* `immigration_sevent` *from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_immigration(db = getOption("RRDdb", "LEEF.RRD.sqlite"))
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

Value

Link to the table `density` in the RRD to be used with `dplyr` and friends

---

`db_read_light_decline` *Read* `light_decline` *from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_light_decline(db = getOption("RRDdb", "LEEF.RRD.sqlite"))
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

Value

Link to the table `density` in the RRD to be used with `dplyr` and friends

---

`db_read_o2` *Read* `o2` *from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_o2(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  from_timestamp = 20210920,
  to_timestamp = 21000101
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`

Value

Link to the table `density` in the RRD to be used with `dplyr` and friends

---

`db_read_stressor_levels` *Read* `stressor_levels` *from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_stressor_levels(db = getOption("RRDdb", "LEEF.RRD.sqlite"))
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

Value

Link to the table `stressor_levels` in the RRD to be used with `dplyr` and friends

---

`db_read_table` *Read a table from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_table(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  table = NULL,
  quiet = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`table`	the name of the table. If ‘NULL‘, a list of tables in the database ‘db‘ will be returned.
`quiet`	if `TRUE` a warning will be issued whe the `table` name is NULL.

Value

Link to the table `density` in the RRD to be used with `dplyr` and friends

---

`db_read_toc` *Read* `toc` *measurements from RRD database and return as an* `tbl_SQLiteConnection`.

Description

This function does not actually fetch the data, but returns an `tbl_SQLiteConnection` object which can be further processed / filtered using `dplyr`. See https://db.rstudio.com/r-packages/dplyr/ for details. To fetch the actual data, execute `collect()`.

Usage

```
db_read_toc(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  from_timestamp = 20210920,
  to_timestamp = 21000101,
  duplicates = mean
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`from_timestamp`	`integer`. Earliest timestamp to return. Default: `20210920`
`to_timestamp`	`integer`. Latest timestamp to return. Default: `21000101`
`duplicates`	function which will be used to combine duplicates. If `NULL`, co summarising is done. Default is ‘mean‘.

Value

Link to the table `density` in the RRD to be used with `dplyr` and friends

---

`density_flowcytometer_archive`

Gate and extract densities from flowcytometer data by using the archived data

---

Description

Gate and extract densities from flowcytometer data by using the archived data

Usage

```
density_flowcytometer_archive(
  extracted_dir =
   "/Volumes/LEEF-1_archive/LEEF.archived.data/LEEF/3.archived.data/extracted/",
  gates_coordinates,
  timestamps,
  output,
  use_H,
  log10_all = FALSE,
  min_FSC.A = NULL,
  particles = NULL,
  mc.cores = 1
)
```

Arguments

`extracted_dir`	srchive directory of the extracted data
`gates_coordinates`	the `gates_coordinates`
`timestamps`	`character` vector containing the timestamps to be classified
`output`	path to which the classified data will be saved as `rds`
`use_H`	if `TRUE`, gating will be done using `height`, otherwie `area`
`log10_all`	if `TRUE`, all data not yet log10 transformed will be log10 transformed ("FL2-A", "FL1-H", "FL2-H", "FL3-H", "FL4-H", "FSC-H", "SSC-H") in the same way as in the pipeline.
`min_FSC.A`	numeric. If `!NULL`, `FSA.A` <= `min_FSC.A` will be fitered out by using a rectangular filter `flowCore::rectangleGate(filterId="filter_out_0", "FSC-A" = c(min_FSC.A, +Inf))`
`particles`	particle class to extract. Mainly `bacteria` or `algae`.
`mc.cores`	number of cores to be used. Defaults to 1

Value

invisible `NULL`

`extract_timestamps` *Extract data from table*

Description

Extract data from `table` matching the `timestamps`. For each `timestamp` the data will be saved as an `rds` file. If `delete_data == TRUE` the data will be deleted from the database.

Usage

```
extract_timestamps(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  table = NULL,
  timestamps,
  delete_data = FALSE,
  only_delete = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`table`	the name of the table. If `NULL`, a list of tables in the database `db` will be shown.
`timestamps`	timestamps which should be extracted
`delete_data`	**Attention!** If `TRUE` the data is deleted from the database!
`only_delete`	if `TRUE` data will only be deleted and not extracted.

Value

invisibly NULL

`extract_traits_flowcytometer_archive` *Extract traits from flowcytometer data by using the archived data*

Description

Extract traits from flowcytometer data by using the archived data

Usage

```
extract_traits_flowcytometer_archive(
  extracted_dir =
   "~/Desktop/flowcytometer.FIXED/LEEF.FIXED.archived.data/LEEF/3.archived.data/extracted/",
  gates_coordinates,
  particles = "bacteria",
  timestamps,
  output,
  length_slope,
  length_intercept,
  use_H,
  min_FSC.A,
  log10_all = FALSE,
  mc.cores = 1,
  wellid_keyword = "$WELLID"
)
```

Arguments

`gates_coordinates`	the `gates_coordinates`
`particles`	particle class to extract. Mainly `bacteria` or `algae`, See `LEEF.measurement.flowcytometer::extract` for details.
`timestamps`	`character` vector containing the timestamps to be classified
`output`	path to which the classified data will be saved as `rds`
`length_slope`	slope of the linear regression of FSC.A and size (lm(mean_FSC.A ~ diameter_micrometer)
`length_intercept`	intercept of the linear regression of FSC.A and size (lm(mean_FSC.A ~ diameter_micrometer)
`use_H`	if `TRUE`, gating will be done using `height`, otherwie `area`
`min_FSC.A`	numeric. If `!NULL`, `FSA.A` <= `min_FSC.A` will be fitered out by using a rectangular filter `flowCore::rectangleGate(filterId="filter_out_0", "FSC-A" = c(min_FSC.A, +Inf))`
`log10_all`	if `TRUE`, all data not yet log10 transformed will be log10 transformed ("FL2-A", "FL1-H", "FL2-H", "FL3-H", "FL4-H", "FSC-H", "SSC-H") in the same way as in the pipeline.
`mc.cores`	number of cores to be used. Defaults to 1
`wellid_keyword`	the kwyword which is used to identify the well ID. Usually "$WELLID" (default), but for the EAWAG Flowcytometer it is "$SMNO".

Value

invisible `NULL`

`generate_overlays` *Just a function to update the overlays.*

Description

Just a function to update the overlays.

Usage

```
generate_overlays(
  params = list(cores = 7, pre_processed_folder =
   "/Volumes/LEEF-1_archive/LEEF.archived.data/LEEF/3.archived.data/pre_processed/",
     extracted_dir =
   "/Volumes/LEEF-1_archive/LEEF.archived.data/LEEF/3.archived.data/extracted/",
     output_dir = "./overlays/"),
  overwrite = FALSE
)
```

Arguments

`params`	list of parameter
`overwrite`	if ‘TRUE‘, existing overlays will be ov erwritten. Default is ‘FALSE‘

`LEEF_1_biomass_bemovi_16` *LEEF-1 - Add biomass to traits*

Description

LEEF-1 - Add biomass to traits

Usage

```
LEEF_1_biomass_bemovi_16(ciliate_traits_16, ciliate_density_16)
```

Arguments

`ciliate_traits_16`	traits as read from file `morph_mvt_TIMESTAMP.rds`
`ciliate_density_16`	density as read from file `mean_density_per_ml_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_1_biomass_bemovi_25`

LEEF-1 - Add biomass to traits

Description

LEEF-1 - Add biomass to traits

Usage

```
LEEF_1_biomass_bemovi_25(ciliate_traits_25, ciliate_density_25)
```

Arguments

`ciliate_traits_25`	traits as read from file `morph_mvt_TIMESTAMP.rds`
`ciliate_density_25`	density as read from file `mean_density_per_ml_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_1_biomass_bemovi_25_cropped`

LEEF-1 - Add biomass to traits

Description

LEEF-1 - Add biomass to traits

Usage

```
LEEF_1_biomass_bemovi_25_cropped(ciliate_traits_25, ciliate_density_25)
```

Arguments

`ciliate_traits_25`	traits as read from file `morph_mvt_TIMESTAMP.rds`
`ciliate_density_25`	density as read from file `mean_density_per_ml_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_1_biomass_bemovi_25_non_cropped`

LEEF-1 - Add biomass to traits

Description

LEEF-1 - Add biomass to traits

Usage

```
LEEF_1_biomass_bemovi_25_non_cropped(ciliate_traits_25, ciliate_density_25)
```

Arguments

`ciliate_traits_25`	traits as read from file `morph_mvt_TIMESTAMP.rds`
`ciliate_density_25`	density as read from file `mean_density_per_ml_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_1_biomass_flowcam`

LEEF-1 - Add biomass to traits

Description

LEEF-1 - Add biomass to traits

Usage

```
LEEF_1_biomass_flowcam(algae_traits, algae_density)
```

Arguments

`algae_traits`	algae traits as read from file `algae_traits_TIMESTAMP.rds`
`algae_density`	algae density as read from file `algae_density_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_1_biomass_flowcytometer`

LEEF-1 - Add biomass to traits

Description

LEEF-1 - Add biomass to traits

Usage

```
LEEF_1_biomass_flowcytometer(traits, density)
```

Arguments

`ciliate_traits_16`	traits as read from file `morph_mvt_TIMESTAMP.rds`
`ciliate_density_16`	density as read from file `mean_density_per_ml_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_2_add_experiment_tables`

Add `composition` *and* `experimetal_design` *table to db.*

Description

Add `composition` and `experimetal_design` table to db.

Usage

```
LEEF_2_add_experiment_tables(
  db = getOption("RRDdb", "LEEF-2.RRD.sqlite"),
  composition = NULL,
  experimetal_design = NULL,
  stressor_levels = NULL,
  immigration_schedule = NULL,
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`composition`	fully qualified path to the `csv` file with the composition data
`stressor_levels`	fully qualified path to the `csv` file with the stressor levels for each sampling date
`immigration_schedule`	fully qualified path to the `csv` file with the immigration event dates
`overwrite`	if `TRUE`, the existing tables will be overwritten. There is actually no need for this.

`LEEF_2_biomass_bemovi` *LEEF-1 - Add biomass to traits*

Description

LEEF-1 - Add biomass to traits

Usage

```
LEEF_2_biomass_bemovi(
  ciliate_traits,
  ciliate_density,
  extrapolation_factor,
  cropping_factor
)
```

Arguments

`ciliate_density`	density as read from file `mean_density_per_ml_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_2_biomass_flowcam` *LEEF-2 - Add biomass to traits*

Description

LEEF-2 - Add biomass to traits

Usage

```
LEEF_2_biomass_flowcam(algae_traits, algae_density)
```

Arguments

`algae_traits`	algae traits as read from file `algae_traits_TIMESTAMP.rds`
`algae_density`	algae density as read from file `algae_density_TIMESTAMP.rds`

Value

list containing two objects, `traits` containing complete traits file as the argument `algai_traits` day includinc biomass column and `biomasses` per timestamp, bottle and species per milliliter.

`LEEF_2_classify_bemovi_archive` *Classify bemovi data by using the archived data*

Description

Classify bemovi data by using the archived data

Usage

```
LEEF_2_classify_bemovi_archive(
  extracted_dir = NULL,
  magnification = 16,
  bemovi_extract_name = NULL,
  timestamps,
  classifier = NULL,
  output,
  exclude_videos = NULL,
  mc.cores = 1
)
```

Arguments

`magnification`	the magnification or the bemovi videos to be reclassified
`bemovi_extract_name`	the name of the `.yml` containing the parameter for the analysis in the directory of the data. The directory depends on the `root_dir`, `magnification` and `timestamp`
`timestamps`	`character` vector containing the timestamps to be classified
`classifier`	classifier for bemovi 16 including path
`output`	path to which the classified data will be saved as `rds`
`exclude_videos`	file names of videos to exclude. If `NULL`, all will be used.
`mc.cores`	number of cores to be used. Defaults to 1
`directory`	with extracted data

Value

invisible `NULL`

`LEEF_2_classify_bemovi_files`

Classify algae_traits data.frame

Description

Classify algae_traits data.frame

Usage

```
LEEF_2_classify_bemovi_files(
  datadir,
  bemovi_extract_name = NULL,
  classifier = NULL,
  exclude_videos = NULL
)
```

Arguments

`datadir`	`character` vector containing the root directory for all files
`bemovi_extract_name`	`character` vector containing the name of the bemovi config file including path.
`classifier`	classifier for bemovi 16 including path
`exclude_videos`	file names of videos to exclude. If `NULL`, all will be used.

Value

`list` containing three objects: - `mean_density_per_ml` - `morph_mvt` - `trajectory_data`

`LEEF_2_classify_flowcam_archive` *Classify algae_traits data.frame*

Description

Classify algae_traits data.frame

Usage

```
LEEF_2_classify_flowcam_archive(
  extracted_dir = NULL,
  timestamps,
  algae_traits_name = "algae_traits_filtered.rds",
  classifier = NULL,
  species_tracked = NULL,
  output,
  mc.cores = 1,
  bottle = NULL
)
```

Arguments

`extracted_dir`	path to the extracted data
`timestamps`	`character` vector containing the timestamps to be classified
`classifier`	classifier for flowcytometer
`species_tracked`	names of the species tracked as a character vector. If `NULL` it will be read from the original configuration file in the `datadir`.
`output`	path to which the classified data will be saved as `rds`
`mc.cores`	number of cores to be used. Defaults to 1
`bottle`	if not 'NULL' (default) only classify this bottle. Needs to be a single bottle!

Value

invisible `NULL`

`LEEF_2_classify_flowcam_files` *Classify algae_traits data.frame*

Description

Classify algae_traits data.frame

Usage

```
LEEF_2_classify_flowcam_files(
  datadir,
  algae_traits_name = "algae_traits_filtered.rds",
  classifier = NULL,
  timestamp = "55555555",
  species_tracked = NULL,
  bottle = NULL
)
```

Arguments

`datadir`	`character` vector containing the root directory for all files
`algae_traits_name`	`character` vector containing the name of the algae traits file without path
`classifier`	classifier for flowcam
`timestamp`	timestamp to be used to stamp the classified data
`species_tracked`	names of the species tracked as a character vector. If `NULL` it will be read from the original configuration file in the `datadir`.
`bottle`	if not 'NULL' (default) only classify this bottle. Needs to be a single bottle!

Value

`list` containing two objects: - `algae_traits` including species - `algae_densities` densities of the different particles identified

---

`LEEF_2_density_flowcytometer_archive` *Gate and extract densities from flowcytometer data by using the archived data*

Description

Gate and extract densities from flowcytometer data by using the archived data

Usage

```
LEEF_2_density_flowcytometer_archive(
  extracted_dir,
  timestamps,
  output,
  use_H,
  gates_coordinates,
  min_FSC.A = NULL,
  particles = NULL,
  mc.cores = 1
)
```

Arguments

`extracted_dir`	srchive directory of the extracted data
`timestamps`	`character` vector containing the timestamps to be classified
`output`	path to which the classified data will be saved as `rds`
`use_H`	if `TRUE`, gating will be done using `height`, otherwie `area`
`gates_coordinates`	the `gates_coordinates`
`min_FSC.A`	numeric. If `!NULL`, `FSA.A` <= `min_FSC.A` will be fitered out by using a rectangular filter `flowCore::rectangleGate(filterId="filter_out_0", "FSC-A" = c(min_FSC.A, +Inf))`
`particles`	particle class to extract. Mainly `bacteria` or `algae`.

Value

invisible `NULL`

`LEEF_2_extract_traits_flowcytometer_archive`

Extract traits from flowcytometer data by using the archived data

---

Description

Extract traits from flowcytometer data by using the archived data

Usage

```
LEEF_2_extract_traits_flowcytometer_archive(
  extracted_dir =
   "~/Desktop/flowcytometer.FIXED/LEEF.FIXED.archived.data/LEEF/3.archived.data/extracted/",
  gates_coordinates,
  particles = "bacteria",
  timestamps,
```

```
  output,
  length_slope,
  length_intercept,
  use_H,
  min_FSC.A,
  log10_all = FALSE,
  wellid_keyword = "$WELLID",
  mc.cores = 1
)
```

Arguments

`gates_coordinates`	the `gates_coordinates`
`particles`	particle class to extract. Mainly `bacteria` or `algae`, See `LEEF.measurement.flowcytometer::extract` for details.
`timestamps`	`character` vector containing the timestamps to be classified
`output`	path to which the classified data will be saved as `rds`
`length_slope`	slope of the linear regression of FSC.A and size (lm(mean_FSC.A ~ diameter_micrometer)
`length_intercept`	intercept of the linear regression of FSC.A and size (lm(mean_FSC.A ~ diameter_micrometer)
`use_H`	if `TRUE`, gating will be done using `height`, otherwie `area`
`min_FSC.A`	numeric. If `!NULL`, `FSA.A` <= `min_FSC.A` will be fitered out by using a rectangular filter `flowCore::rectangleGate(filterId="filter_out_0", "FSC-A" = c(min_FSC.A, +Inf))`
`log10_all`	if `TRUE`, all data not yet log10 transformed will be log10 transformed ("FL2-A", "FL1-H", "FL2-H", "FL3-H", "FL4-H", "FSC-H", "SSC-H") in the same way as in the pipeline.
`wellid_keyword`	the kwyword which is used to identify the well ID. Usually "$WELLID" (default), but for the EAWAG Flowcytometer it is "$SMNO".
`mc.cores`	number of cores to be used. Defaults to 1

Value

invisible `NULL`

`LEEF_2_make_view_density`

Create `density` *view wgich contains all density data from all measurements for LEEF-2*

Description

Create `density` view wgich contains all density data from all measurements for LEEF-2

Usage

```
LEEF_2_make_view_density(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2022-11-07",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`LEEF_2_make_view_o2` *Create* `o2` *view which contains selected fields from the os measurement for LEEF-2*

Description

Create `o2` view which contains selected fields from the os measurement for LEEF-2

Usage

```
LEEF_2_make_view_o2(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2022-11-07",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`LEEF_2_make_view_toc` *Create* `toc` *view wich contains selected fields from the toc__toc measurement for LEEF-2*

Description

Create `toc` view wich contains selected fields from the toc__toc measurement for LEEF-2

Usage

```
LEEF_2_make_view_toc(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2022-11-07",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`LEEF_2_plot_bottles_per_timestamp` *Plot diagnostic plot to check bottles per timestamp*

Description

Plot diagnostic plot to check bottles per timestamp

Usage

```
LEEF_2_plot_bottles_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  lastDays = 7,
  arrow = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`lastDays`	last days to plot. Defaoult `7`
`arrow`	if `TRUE` read data from arrow instead of sqlite database

Value

`ggplot` object of the plot

`LEEF_2_plot_conductivity_per_bottle_per_timestamp` *Plot diagnostic plot to check number of species per timestamp*

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
LEEF_2_plot_conductivity_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  treatment_begin_day = 70,
  treatment_end_day = 154,
  arrow = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`treatment_begin_day`	begin of treatment (vertical red line in plot). If `NULL` none is plotted.
`treatment_end_day`	end of treatment (vertical red line in plot). If `NULL` none is plotted.
`arrow`	if `TRUE` read data from arrow instead of sqlite database

Value

`ggplot` object of the plot

`LEEF_2_plot_density_species_per_bottle_per_timestamp` *Plot diagnostic plot to check number of species per timestamp for LEEF-2*

Description

Plot diagnostic plot to check number of species per timestamp for LEEF-2

Usage

```
LEEF_2_plot_density_species_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  transform_density_4throot = TRUE,
  measurement = "bemovi_mag_16",
  species_set_id = NULL,
  treatment_begin_day = 70,
  treatment_end_day = 154,
  arrow = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`transform_density_4throot`	if `TRUE`, density is transformed using 4th root transformation.
`measurement`	the measurement to be plotted. If `NULL`, the default, they are plotted by temperature treatment (constant & increasing)
`species_set_id`	id of species set to use for filtering
`treatment_begin_day`	begin of treatment (vertical red line in plot). If `NULL` none is plotted.
`treatment_end_day`	end of treatment (vertical red line in plot). If `NULL` none is plotted.
`arrow`	if `TRUE` read data from arrow instead of sqlite database

Value

`ggplot` object of the plot

`LEEF_2_plot_flowcytometer_individual_per_bottle_per_timestamp` *Plot diagnostic plot to check is plte has an impact on flowcytometer for LEEF-2*

Description

Plot diagnostic plot to check is plte has an impact on flowcytometer for LEEF-2

Usage

```
LEEF_2_plot_flowcytometer_individual_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  species = c("bacteria"),
  arrow = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`species`	species to be plotted
`arrow`	if `TRUE` read data from arrow instead of sqlite database

Value

`ggplot` object of the plot

`LEEF_2_plot_o2s_per_bottle_per_timestamp` *Plot diagnostic plot to check number of species per timestamp*

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
LEEF_2_plot_o2s_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  treatment_begin_day = 70,
  treatment_end_day = 154,
  arrow = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`treatment_begin_day`	begin of treatment (vertical red line in plot). If `NULL` none is plotted.
`treatment_end_day`	end of treatment (vertical red line in plot). If `NULL` none is plotted.
`arrow`	if `TRUE` read data from arrow instead of sqlite database

Value

`ggplot` object of the plot

`LEEF_2_plot_response_incubator`

Plot diagnostic plot to check number of species per timestamp

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
LEEF_2_plot_response_incubator(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  arrow = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`arrow`	if `TRUE` read data from arrow instead of sqlite database

Value

`ggplot` object of the plot

`LEEF_2_plot_tocs_per_bottle_per_timestamp` *Plot diagnostic plot to check number of species per timestamp*

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
LEEF_2_plot_tocs_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  type = c("IC", "TC", "TN", "TOC"),
  treatment_begin_day = 70,
  treatment_end_day = 154,
  arrow = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`type`	the type of the measurement which should be displayed. A vector with the types. Possible values are: "TOC", "TN", "IC", "TN", "".
`treatment_begin_day`	begin of treatment (vertical red line in plot). If `NULL` none is plotted.
`treatment_end_day`	end of treatment (vertical red line in plot). If `NULL` none is plotted.
`arrow`	if `TRUE` read data from arrow instead of sqlite database

Value

`ggplot` object of the plot

`LEEF_2_rename_composition` *FUNCTION_TITLE*

Description

FUNCTION_DESCRIPTION

Usage

```
LEEF_2_rename_composition(object)
```

Arguments

`object`	PARAM_DESCRIPTION

Details

DETAILS

Value

OUTPUT_DESCRIPTION

Examples

```
## Not run:
if (interactive()) {
  # EXAMPLE1
}

## End(Not run)
```

`LEEF_2_rename_species` *FUNCTION_TITLE*

Description

FUNCTION_DESCRIPTION

Usage

```
LEEF_2_rename_species(x)
```

Arguments

`x`	PARAM_DESCRIPTION

Details

DETAILS

Value

OUTPUT_DESCRIPTION

Examples

```
## Not run:
if (interactive()) {
  # EXAMPLE1
}

## End(Not run)
```

`LEEF_2_rename_species_prob_columns`

FUNCTION_TITLE

Description

FUNCTION_DESCRIPTION

Usage

```
LEEF_2_rename_species_prob_columns(object)
```

Arguments

`object`	PARAM_DESCRIPTION

Details

DETAILS

Value

OUTPUT_DESCRIPTION

Examples

```
## Not run:
if (interactive()) {
  # EXAMPLE1
}

## End(Not run)
```

`LEEF_2_rename_toc` *FUNCTION_TITLE*

Description

FUNCTION_DESCRIPTION

Usage

```
LEEF_2_rename_toc(x)
```

Arguments

`x`	PARAM_DESCRIPTION

Details

DETAILS

Value

OUTPUT_DESCRIPTION

Examples

```
## Not run:
if (interactive()) {
  # EXAMPLE1
}

## End(Not run)
```

`LEEF_2_report_diagnostic` *Report for checking pipeline and sampling*

Description

Report for checking pipeline and sampling

Usage

```
LEEF_2_report_diagnostic(
  db = NULL,
  suffix = "",
  format = "html",
  lastDays = 7,
  arrow = FALSE,
  parquet_dir = "/Volumes/RRD.Reclassification_LEEF-2/parquet/"
)
```

Arguments

`db`	fully qualified path to the sqlite database. The report will be saved to the same directory.
`suffix`	suffix for the file name
`format`	the format of the report as a character vector of length 1. Supported are at the moment: `html` (the default), `pdf` and `word`.
`lastDays`	number of last days to be included in graph 2. Default: 7
`arrow`	if `TRUE` read data from arrow instead of sqlite database
`parquet_dir`	directory where the parquet files are stored. Default: '/Volumes/RRD.Reclassification_LEEF-2/parquet/'. Will only be used whe `arrow` = `TRUE`

Value

the fully qualified file name to the created report.

`make_views` *Create all views*

Description

This function calls the other `make_view_...()` functions, wrapped in individual `try()` blocks.

Usage

```
make_views(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2021-09-20",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment in the format `YYYY-MM-DD`
`overwrite`	if `TRUE`, overwrite existing view

Value

invisibly `NULL`

`make_view_conductivity` *Create `o2` view wgich contains selected fields from the conductivity measurement*

Description

Create `o2` view wgich contains selected fields from the conductivity measurement

Usage

```
make_view_conductivity(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2021-09-20",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`make_view_density` *Create `density` view wgich contains all density data from all measurements*

Description

Create `density` view wgich contains all density data from all measurements

Usage

```
make_view_density(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2021-09-20",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`make_view_immigration_event`

Create `immigration_event` *view which adds the experimental day*

Description

Create `immigration_event` view which adds the experimental day

Usage

```
make_view_immigration_event(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2021-09-20",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`make_view_light_decline` *Create* `light_decline` *view which adds the experimental day*

Description

Create `light_decline` view which adds the experimental day

Usage

```
make_view_light_decline(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2021-09-20",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the creation of the view.

`make_view_o2` *Create* `o2` *view wgich contains selected fields from the os measurement*

Description

Create `o2` view wgich contains selected fields from the os measurement

Usage

```
make_view_o2(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2021-09-20",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`make_view_toc` *Create* `toc` *view wich contains selected fields from the toc__toc measurement*

Description

Create `toc` view wich contains selected fields from the toc__toc measurement

Usage

```
make_view_toc(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  start_date = "2021-09-20",
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`start_date`	the start date of the experiment
`overwrite`	if `TRUE`, overwrite existing view

Value

the result of the execution of the ecreatiuon of the view.

`overlays` *Create overlays*

Description

Create overlays

Usage

```
overlays(
  timestamp = 20210920,
  magnification = 25,
  cropped = FALSE,
  label = "species",
  overlay.type = "both",
  crf = 17,
  gamma = 2,
  ffmpeg = "ffmpeg",
  from_current_to_archive_dir =
    "./../../../../../Duck/LEEFSwift3/LEEF.archived.data/LEEF/3.archived.data",
  mc.cores = 1
)
```

Arguments

`timestamp`	the timestamp of the sampling day
`magnification`	the magnification of the bemovi method
`cropped`	if `TRUE`, the cropped measurement should be used
`label`	column to be used to label the particle. Default `"pecies"`.
`overlay.type`	Overlays can either be shown as "label", "circle" or "both". Default: `"both"`
`crf`	integer value between 1 to 51, where 1 means lossless, 17 is nearly visually lossless, 51 is worst quality. Default value is 23
`gamma`	increase video dynamic range. Value between 0.1 and 10. Default 2. see `https://ffmpeg.org/ffmpeg-filters.html#eq` for further info
`ffmpeg`	command to run ffmpeg. It can include a path. Default `ffmpeg`
`from_current_to_archive_dir`	path from the current directory to the archived data can be found and the folders `extracted` and `pre-processed`
`mc.cores`	number of cores to be used for parallel execution. Defaults to 1

`overlays_from_folders` *Create overlays*

Description

This is a wrapper around the function `bemovi.LEEF::create_overlays_subtitle_directory()` providing default values for LEEF. Overlays will be created from a folder containing the video files and the bemovi config file and the trajectory data file name.

Usage

```
overlays_from_folders(
  traj_data_file,
  avi_folder,
  bemovi_extract_yml_file,
  temp_overlay_folder,
  overlay_folder,
  overlay_type = "both",
  label = "species",
  ffmpeg = "ffmpeg",
  font_size = 24,
  circle_size = 120,
  crf = 23,
  gamma = 2,
  mc_cores = 1,
  overwrite = FALSE
)
```

Arguments

`traj_data_file`	file name of the file containing the trajectory data (usually from the Master file)
`avi_folder`	folder containing the input `.avi` files
`bemovi_extract_yml_file`	name of the `bemovi_extract.yml` config file
`temp_overlay_folder`	directory where the subtitle files will be saved
`overlay_folder`	directory where the overlay video will be saved
`overlay_type`	option for the overlays. Overlays can either be shown as `"label"`, `"circle"` or `"both"`
`label`	column to be used to label the particle. Default is `"trajectory"`, other useful might be `"species"`
`ffmpeg`	command to run ffmpeg. The default is `par_ffmpeg()`. It can include a path.
`font_size`	size of the font for the labels. Default: 24
`circle_size`	size of the circle. Default: 120

`crf`	integer value between 1 to 51, where 1 means lossless, 17 is nearly visually lossless, 51 is worst quality. Default value is 23
`gamma`	gamma correction. Value between 0.1 and 10. Default 2. see https://ffmpeg.org/ffmpeg-filters.html#eq for further info
`mc_cores`	number of cores toi be used for parallel execution. Defaults to `par_mc.cores()`
`overwrite`	if `TRUE`, existing overlays will be overwritten. Default is `FALSE`

`overlays_from_reclassification`

Create overlays

Description

Create overlays

Usage

```
overlays_from_reclassification(
  archive_dir = "/Users/rainerkrug/Duck/LEEFSwift3",
  classification_dir = "~/RRD.Reclassification_1",
  magnification = 16,
  yml_postfix = NULL,
  timestamp,
  overlay_folder = NULL,
  overlay_type = "both",
  label = "species",
  ffmpeg = "ffmpeg",
  font_size = 24,
  circle_size = 120,
  crf = 23,
  gamma = 2,
  mc_cores = 1
)
```

Arguments

`magnification`	the magnification of the bemovi method
`timestamp`	the timestamp of the sampling day
`label`	column to be used to label the particle. Default `ffmpeg`
`ffmpeg`	command to run ffmpeg. It can include a path. Default
`crf`	integer value between 1 to 51, where 1 means lossless, 17 is nearly visually lossless, 51 is worst quality. Default value is 23
`gamma`	increase video dynamic range. Value between 0.1 and 10. Default `"pecies"` 2. see https://ffmpeg.org/ffmpeg-filters.html#eq for further info
`mc_cores`	number of cores to use. Default `1`

`parquet_add_bemovi_16` *FUNCTION_TITLE*

Description

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

FUNCTION_DESCRIPTION

Essentially combined re-implementation of `rds_to_parquet` and `csv_to_parquet` plus standardisation on small letter column names and timestamps as character.

Usage

```
parquet_add_bemovi_16(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

parquet_add_bemovi_25(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

parquet_add_bemovi_25_cropped(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

parquet_add_flowcam(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
```

```
  rename = FALSE
)

parquet_add_flowcytometer(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

parquet_add_o2(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

parquet_add_manualcount(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

parquet_add_conductivity(
  fns = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

parquet_add_toc(
  fn = NULL,
  path_to_parquet_root_dir = NULL,
  compression = "snappy",
  rename = FALSE
)

object_to_parquet(
  object,
  path_to_parquet,
  partition = "no",
  compression = "snappy",
  compression_level = NULL,
  ...
)
```

Arguments

`fns`	PARAM_DESCRIPTION, Default: NULL
`path_to_parquet_root_dir`	PARAM_DESCRIPTION, Default: NULL
`compression`	compression as used in `write_parquet`, Default: "snappy"
`rename`	Should data be renamed to follow specified standards, Default: FALSE
`fn`	PARAM_DESCRIPTION, Default: NULL
`object`	object to be written as a parquet file / arrow db
`path_to_parquet`	PARAM_DESCRIPTION, Default: NULL
`compression_level`	PARAM_DESCRIPTION, Default: "NULL"

Details

DETAILS

DETAILS

DETAILS

DETAILS

DETAILS

DETAILS

DETAILS

DETAILS

DETAILS

DETAILS

Value

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

OUTPUT_DESCRIPTION

See Also

pbapply

pbapply

pbapply

pbapply

pbapply

pbapply

pbapply

Examples

```
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
```

```
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
## Not run:
if (interactive()) {
    # EXAMPLE1
}

## End(Not run)
```

`plot_bottles_per_timestamp`

Plot diagnostic plot to check bottles per timestamp

Description

Plot diagnostic plot to check bottles per timestamp

Usage

```
plot_bottles_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  lastDays = 7
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`lastDays`	last days for which to plot

Value

`ggplot` object of the plot

`plot_density_species_per_bottle_per_timestamp`

Plot diagnostic plot to check number of species per timestamp

---

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
plot_density_species_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  transform_density_4throot = TRUE,
  measurement = "bemovi_mag_16",
  species_set_id = NULL
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`transform_density_4throot`	if `TRUE`, density is transformed using 4th root transformation.
`measurement`	the measurement to be plotted. If `NULL`, the default, they are plotted by temperature treatment (constant & increasing)
`species_set_id`	name of the set of species

Value

`ggplot` object of the plot

`plot_no_species_per_bottle_per_timestamp`

Plot diagnostic plot to check number of species per timestamp

---

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
plot_no_species_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite")
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

Value

`ggplot` object of the plot

`plot_o2s_per_bottle_per_timestamp`

Plot diagnostic plot to check number of species per timestamp

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
plot_o2s_per_bottle_per_timestamp(db = getOption("RRDdb", "LEEF.RRD.sqlite"))
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

Value

`ggplot` object of the plot

`plot_response_incubator`

Plot diagnostic plot to check number of species per timestamp

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
plot_response_incubator(db = getOption("RRDdb", "LEEF.RRD.sqlite"))
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

Value

`ggplot` object of the plot

`plot_species_per_timestamp_per_measure`

Plot diagnostic plot to check species per timestamp per method

Description

Plot diagnostic plot to check species per timestamp per method

Usage

```
plot_species_per_timestamp_per_measure(
  db = getOption("RRDdb", "LEEF.RRD.sqlite")
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

Value

`ggplot` object of the plot

`plot_tocs_per_bottle_per_timestamp`

Plot diagnostic plot to check number of species per timestamp

Description

Plot diagnostic plot to check number of species per timestamp

Usage

```
plot_tocs_per_bottle_per_timestamp(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  type = c("IC", "TC", "TN", "TOC")
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`type`	the type of the measurement which should be displayed. A vector with the types. Possible values are: "TOC", "TN", "IC", "TN", "".

Value

`ggplot` object of the plot

`prepare_rrd` *Prepare the RRD database for usage with the other functions in this package.*

Description

This function adds the needed tables and creates the vuews needed for the analysis.

Usage

```
prepare_rrd(
  db = getOption("RRDdb", "LEEF.RRD.sqlite"),
  composition = NULL,
  experimetal_design = NULL,
  overwrite = FALSE
)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`
`composition`	fully qualified path to the `csv` file with the composition data
`experimetal_design`	fully qualified path to the `csv` file with the experimetal_design data
`overwrite`	if `TRUE`, the existing tables / views will be overwritten.

`read_sql` *Read data from the RRD database*

Description

The function executes the provided sql statement and returns the resulting table0. For security, the database is opened in **read-only** mode.!

Usage

```
read_sql(db = getOption("RRDdb", "LEEF.RRD.sqlite"), sql)
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to `LEEF.RRD.sqlite`
`sql`	sql statement

Value

the table resulting from the query as a `data.frame` object.

`render_quarto_temp` *Render a quarto file in a temporary directory and copy the resulting files back to the working directory*

Description

Render a quarto file in a temporary directory and copy the resulting files back to the working directory

Usage

```
render_quarto_temp(input, output_format = "html", ...)
```

Arguments

`input`	the input qmd file
`output_format`	the output format
`...`	additional arguments for `quarto::quarto_render()`

`report_diagnostic` *Report for checking pipeline and sampling*

Description

Report for checking pipeline and sampling

Usage

```
report_diagnostic(
  db,
  template = "LEEF_1",
  suffix = "",
  format = "html",
  lastDays = 7
)
```

Arguments

`db`	fully qualified path to the sqlite database. The report will be saved to the same directory.
`template`	Template to be used for report. At the moment only `"LEEF_1}` and `\code{"LEEF_2` supported.
`suffix`	suffix for the file name
`format`	the format of the report as a character vector of length 1. Supported are at the moment: `html` (the default), `pdf` and `word`.
`lastDays`	number of last days to be included in graph 2. Default: 7

Value

the fully qualified file name to the created report.

`report_flowcytometer` *Report for checking pipeline and sampling*

Description

Report for checking pipeline and sampling

Usage

```
report_flowcytometer(
  timestamp = "20230106",
  extracted_base_dir =
    "~/Duck/LEEFSwift3/LEEF_2.archived.data/LEEF/3.archived.data/extracted/",
  leef = "LEEF-2",
  output_dir = ".",
  format = "html",
  browse = TRUE
)
```

Arguments

`timestamp`	one or more timestamp to be plotted. If `NULL`, all timestamps available in `extracted_base_dir` will be plotted
`extracted_base_dir`	directory in which the extracted data can be found with filenames as in the archive
`leef`	LeEEF experiment, either `"LEEF-1"` or `"LEEF-2"`
`output_dir`	output directory of the fil=nal report
`format`	the format of the report as a character vector of length 1. Supported are at the moment: `html` (the default), `pdf` and `word`.
`browse`	if `TRUE` (the default) opr=en the report in the appropriate program.

Value

the fully qualified file name to the created report.

`report_interactive` *Run interactive report*

Description

Run interactive report

Usage

```
report_interactive(db)
```

Arguments

`db`	fully qualified path to the sqlite database. The report will be saved to the same directory.

Value

invisibly `NULL`

`report_reclassification` *Do the reclassification and create a report*

Description

The report needs to adapted to new classifications.

Usage

```
report_reclassification(
  reclassification_report = system.file(leef, "Reclassification_20220105.Rmd", package =
    "LEEF.analysis"),
  leef = "LEEF-1",
  output_dir = normalizePath(file.path("~", "Reclassification_20220105")),
  output_name = "Reclassification_20220105",
  format = "html"
)
```

Arguments

`reclassification_report`	The name of the reclassification report. The default points to the first one. Included reclassification reports are: • `system.file(Reclassification_2022.01.10.Rmd, package = "LEEF.analysis")` • `system.file(Reclassification_20220105_20220221_flowcam.Rmd, package = "LEEF.analysis")` • `system.file(Reclassification_20220506.Rmd, package = "LEEF.analysis")`
`leef`	LEEF study - allowed are `"LEEF-1` and `"LEEF-2"`
`output_dir`	directory of the output of the reclassification and the report
`output_name`	The name of the final report, without extension.
`format`	the format of the report as a character vector of length 1. Supported are at the moment: `html` (the default), `pdf` and `word`.

Value

the fully qualified file name to the created report.

`report_sanity_check` *Report for the sanity check of the data for submission to the pipeline*

Description

Report for the sanity check of the data for submission to the pipeline

Usage

```
report_sanity_check(sample_dir, suffix = "", format = "html")
```

Arguments

`sample_dir`	root dir in which the folder `00.general.parameter` and `0.raw.data` are located
`suffix`	suffix for the file name
`format`	the format of the report as a character vector of length 1. Supported are at the moment: `html` (the default), `pdf` and `word`.

Value

the fully qualified file name to the created report.

`RRD_create_indices` *Create indices in RRD database*

Description

Create indices in RRD database

Usage

```
RRD_create_indices(dbname, LEEF = NULL, continue_after_error = FALSE)
```

Arguments

`dbname`	the path and name of the database. Must exist!
`either`	"LEEF-1" or "LEEF-2"
`contimue_after_error`	Do not quit execution of sql statements when error occurs but continue. **Use with caution!** Default: `FALSE`

`RRD_LEEF_1_rename` *Rename columns and species in the RRD*

Description

Rename columns and species in the RRD

Usage

```
RRD_LEEF_1_rename(db = getOption("RRDdb", "LEEF.RRD.sqlite"))
```

Arguments

`db`	fully qualified path to the sqlite database. Default, read from option `RRDdb`. If not set, defaults to option `RRDdb`; if this is not set, defaults to `LEEF.RRD.sqlite`

`RRD_new` *Create new RRD database*

Description

Create a new database following the scheme used for the LEEF.RRD database

Usage

```
RRD_new(dbname, LEEF = NULL, DBIDriver = RSQLite::SQLite())
```

Arguments

`dbname`	the path and name of the database. Must not exist.
`DBIDriver`	the DBI driver to use. Default is RSQLite::SQLite()
`either`	"LEEF-1" or "LEEF-2"

`sanity_check` *Sanity check data for pipeline*

Description

Executes onle test per method required.

Usage

```
sanity_check(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `00.general.parameter` and `0.raw.data` are located

Value

named `list` object, with one element per method. If the test passed for the method, the result is `TRUE`, if it fails, a string indicating the reason why it failed

---

`sanity_check_bemovi.mag.16`

Sanity check for bemovi.mag.16

Description

Sanity check for bemovi.mag.16

Usage

```
sanity_check_bemovi.mag.16(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `bemovi.mag.16` and `0.raw.data` are located

Value

`TRUE` if the test passed, a string indicating the error if it failed.

`sanity_check_bemovi.mag.25`

Sanity check for bemovi.mag.25

Description

Sanity check for bemovi.mag.25

Usage

```
sanity_check_bemovi.mag.25(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `bemovi.mag.25` and `0.raw.data` are located

Value

`TRUE` if the test passed, a string indicating the error if it failed.

`sanity_check_flowcam` *Sanity check for flowcam*

Description

Sanity check for flowcam

Usage

```
sanity_check_flowcam(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `flowcam` and `0.raw.data` are located

Value

`TRUE` if the test passed, a string indicating the error if it failed.

`sanity_check_flowcytometer` *Sanity check for flowcytometer*

Description

Sanity check for flowcytometer

Usage

```
sanity_check_flowcytometer(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `flowcam` and `0.raw.data` are located

Value

`TRUE` if the test passed, a string indicating the error if it failed.

`sanity_check_general.data` *Sanity check for 0.general.data*

Description

Sanity check for 0.general.data

Usage

```
sanity_check_general.data(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `00.general.parameter` and `0.raw.data` are located

Value

`TRUE` if the test passed, a string indicating the error if it failed.

`sanity_check_manualcount` *Sanity check for manualcount*

Description

Sanity check for manualcount

Usage

```
sanity_check_manualcount(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `00.general.parameter` and `0.raw.data` are located

Value

`TRUE` if the test passed, a string indicating the error if it failed.

`sanity_check_o2meter` *Sanity check for o2meter*

Description

Sanity check for o2meter

Usage

```
sanity_check_o2meter(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `00.general.parameter` and `0.raw.data` are located

Value

`TRUE` if the test passed, a string indicating the error if it failed.

`sanity_get_timestamp` *Extract timestamp from* `sample_metadata.yml` *file*

Description

Extract timestamp from `sample_metadata.yml` file

Usage

```
sanity_get_timestamp(sample_dir = ".")
```

Arguments

`sample_dir`	root dir in which the folder `00.general.parameter` and `9.raw,data` are located

Value

The extracted timestamp as a string

`SetNotFoundSpeciesTo0` *Set number of Not Found species to 0*

Description

ADD DESCRIPTION

Usage

```
SetNotFoundSpeciesTo0(
  densities,
  mb,
  meas,
  exp_design,
  compositions,
  species.tracked
)
```

Arguments

`species.tracked`

`species_set` *Defines sets of species*

Description

Defines sets of species

Usage

```
species_set(species_set_id = NULL)
```

Arguments

`species_set_id`	if `NULL`, return the names of the species sets defined. If the name or the index of a specific set, return that set.

Value

Names of sets or the set itself

`traits_flowcytometer_archive`

Gate and extract densities from flowcytometer data by using the archived data

---

Description

Gate and extract densities from flowcytometer data by using the archived data

Usage

```
traits_flowcytometer_archive(
  extracted_dir =
   "/Volumes/LEEF-1_archive/LEEF.archived.data/LEEF/3.archived.data/extracted/",
  gates_coordinates,
  timestamps,
  output,
  use_H,
  min_FSC.A,
  log10_all = FALSE,
  particles = c("bacteria", "algae"),
  mc.cores = 1
)
```

Arguments

`extracted_dir`	srchive directory of the extracted data
`gates_coordinates`	the `gates_coordinates`
`timestamps`	`character` vector containing the timestamps to be classified
`output`	path to which the classified data will be saved as `rds`
`use_H`	if `TRUE`, gating will be done using `height`, otherwie `area`
`min_FSC.A`	numeric. If `!NULL`, `FSA.A` <= `min_FSC.A` will be fitered out by using a rectangular filter `flowCore::rectangleGate(filterId="filter_out_0", "FSC-A" = c(min_FSC.A, +Inf))`
`log10_all`	if `TRUE`, all data not yet log10 transformed will be log10 transformed ("FL2-A", "FL1-H", "FL2-H", "FL3-H", "FL4-H", "FSC-H", "SSC-H") in the same way as in the pipeline.
`particles`	the particles, as defined in the gates file, to be extracted. Allowed are one or multiple of `bacteria`, `LNA`, `MNA`, `HNA`, `algae`
`mc.cores`	number of cores to be used. Defaults to 1

Value

invisible `NULL`

Index