Package 'FBtagtools'

June 30, 2021

Title Tools for working with FB SMRT (and Lander2) tag data

Version 0.0.0.9000

Description Tools for working with FB SMRT (and Lander2) tag data

License MIT + file LICENSE

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Depends R (>= 2.10)

Imports googledrive,
dplyr,
stringr

R topics documented:

add_bathy *Add bathymetry data, given locations*

Description

Pulls bathymetry data from either the internet or a provided database and associates it with tag locations. Note: code obtained from Eric Keene (NOAA) via David Sweeney (MarEcoTel) with minor modification. To use, you will need bathymetry data. You can download it from the Drive FreakinBeakinTagData > Acoustic Preliminary Results > Resources-Seafloor.zip or by using `download_bathymetry` (which just locates and downloads that zip file for you).

Usage

```
add_bathy(x, lat_var, lon_var, z_radius = 2.5, bathy_path)
```

Arguments

`x`	A data frame with location data to associate with bathymetry data
`lat_var`	name of variable in `x` with latitude data
`lon_var`	name of variable in `x` with longitude data
`z_radius`	Distance in km around each location that seafloor depth and slope will be calculated. This can be a single value or a vector of different values the same length as the number of rows of the data input. Default: 2.5 km
`bathy_path`	A directory path to the folder containing all bathymetry data. File names in the folder must be like this :'NEPACseafloor-116.000-30.000-115.000-31.000.csv'. If this input is NULL, data will be pulled from online, thus requiring internet access. (Note that there may be some access and speed issues with pulling the data from the web.)

Value

A dataframe with associated bathymetry data as new columns in the dataset

Examples

```
#examples not yet provided, sorry :(
```

add_event_depths *Add time columns to time-series*

Description

Add time (in seconds since tagon and perhaps as datetimes in UTC and/or local timezone) to a tibble or data-frame with regularly sampled data

Usage

```
add_event_depths(x, start_x, z, sampling_rate, start_offset = 0, tagon_time)
```

Arguments

x	input data: vector or data frame with event times as datetimes, or in seconds since tagon. Datetimes are assumed to be UTC.
start_x	name (no quotes needed) of variable in x that gives event times. Not needed if x is a vector or has only one column.
z	depth data: sensor data structure, vector, matrix, tibble, or data frame. If not a sensor structure, then sampling_rate is required.
sampling_rate	sampling rate of depth in Hz (required if x is not a sensor data structure, and ignored if it is)
start_offset	time (in seconds) between start of tag recording and the first recorded sample in z. Ignored (and pulled from metadata) if x is a sensor data structure. default: 0
tagon_time	string giving the tag recording start time. Can be found in tag_dataset$info$dephist_device_datetime_ Format: day month year hours minutes seconds. Required if times in x are datetimes.

Value

A data.frame with the data (one row per event) and additional column depth (in meters) of the animal at the start time of the event. Note: if you want to grab and summarize depth data for longer intervals, consider interval_join or find_dives and dive_stats

Examples

```
Examples will go here
```

---

add_event_times *Add times of events (UTC or seconds_since)*

Description

Add time (in seconds since tagon or datetimes in UTC and/or local timezone) to a tibble or dataframe with event times (conversion between datetime and seconds_since_tagon formats)

Usage

```
add_event_times(x, start_x, tagon_time)
```

Arguments

x	input data frame: vector, tibble, or data frame. There should be one column that gives either seconds_since_tagon or UTC time already present.
start_x	name of the variable in x that gives the times
tagon_time	string giving the tag recording start time (or the info sensor data structure). Can be found in tag_dataset$info$dephist_device_datetime_start. Format: day month year hours minutes seconds.

Value

The input x, but with additional column(s) sec_since_tagon with times in seconds since start of tag recording or utc_time (plus optionally local_time). Note, this function adds UTC times; if you want local times in addition, consider using utc_to_local after this function.

Examples

```
Examples will go here
```

`add_sensor_times` *Add time columns to sensor data time-series*

Description

Add time (in seconds since tagon and perhaps as datetimes in UTC and/or local timezone) to a tibble or data-frame with regularly sampled data

Usage

```
add_sensor_times(
  x,
  sampling_rate,
  start_offset = 0,
  add_utc_time = FALSE,
  tagon_time
)
```

Arguments

`x`	input data: sensor data structure, vector, matrix, tibble, or data frame. If `x` is not a tag sensor data structure, then `sampling_rate` is required input.
`sampling_rate`	sampling rate of x in Hz (required if x is not a sensor data structure, and ignored if it is)
`start_offset`	time (in seconds) between start of tag recording and the first recorded sample. Ignored (and pulled from metadata) if `x` is a sensor data structure. default: 0
`add_utc_time`	Logical: should UTC time column be added? Default is FALSE.
`tagon_time`	string giving the tag recording start time. Can be found in tag_dataset$info$dephist_device_datetime_ Format: day month year hours minutes seconds. Required if `add_dttm` is TRUE.

Value

A data.frame with the data (one row per sample) and column sec_since_tagon with times in seconds since start of tag recording. Optionally, also includes column utc_time.

Examples

```
Examples will go here
```

add_times *Add time columns to time-series*

Description

Add time (in seconds since tagon and perhaps as datetimes in UTC and/or local timezone) to a tibble or data-frame with regularly sampled data

Usage

```
add_times(x, sampling_rate, start_offset = 0, add_utc_time = FALSE, tagon_time)
```

Arguments

x	input data: sensor data structure, vector, matrix, tibble, or data frame. If x is not a tag sensor data structure, then sampling_rate is required input.
sampling_rate	sampling rate of x in Hz (required if x is not a sensor data structure, and ignored if it is)
start_offset	time (in seconds) between start of tag recording and the first recorded sample. Ignored (and pulled from metadata) if x is a sensor data structure. default: 0
add_utc_time	Logical: should UTC time column be added? Default is FALSE.
tagon_time	string giving the tag recording start time. Can be found in tag_dataset$info$dephist_device_datetime_ Format: day month year hours minutes seconds. Required if add_dttm is TRUE.

Value

A data.frame with the data (one row per sample) and column sec_since_tagon with times in seconds since start of tag recording. Optionally, also includes column utc_time.

Examples

```
Examples will go here
```

dive_acoustic_summary *Dive-level acoustic and tag data, for one or more tags*

Description

Summarize SMRT (and/or Lander2) tag data from .nc files for each foraging dive cycle. Note: Currently for only SMRT tags; function and help will be updated to allow inclusion of Lander2 data as well when possible.

Summarize SMRT (and/or Lander2) tag data from .nc files for each foraging dive cycle. Note: Currently for only SMRT tags; function and help will be updated to allow inclusion of Lander2 data as well when possible.

Usage

```
dive_acoustic_summary(
  tag_id = zc_smrt_tag_list,
  nc_path = getwd(),
  ae_path = getwd()
)

dive_acoustic_summary(
  tag_id = zc_smrt_tag_list,
  nc_path = getwd(),
  ae_path = getwd()
)
```

Arguments

tag_id	Character string or vector with tag IDs (without "-cal.nc"). Default: all SMRT ziphius tags.
nc_path	Directory (quoted string) where .nc files are stored. Can be one string, or a list the same length as tag_ids. Note: to download latest versions from Google drive, try function: download_drive_nc. Default: current working directory. Note: use "/" and not "\" to avoid headaches.
ae_path	Directory (quoted string) where text files with info about acoustic events are stored. If needed, you can use download_drive_acoustic_events to get these. Default is the current working directory.
bathy_path	A directory path to the folder containing all bathymetry data. Use download_bathy if you don't have this data already. If not provided, the bathymetry data will not be included in the output dataset.
rl_file	name (with path, if needed) of locally-stored .csv file with raw RL data. If not provided, the default is to use download_drive_rls to obtain it from the FB Google Drive.
ping_log_file	name (with path, if needed) of locally-stored .csv file with raw RL data. If not provided, the default is to use download_drive_rls to obtain it from the FB Google Drive.
email	Email address (required for FB Google Drive authentication; optional if rl_file is provided). You may also be asked to sign in or verify your Google identity as this function runs.
save_csv	Logical; whether or not to save a csv file with the results. Default: FALSE
csv_name	File name (with path, if desired) in which to save results in csv format. Default is acoustic_summary.csv.

Value

A data.frame() with one row per dive, per whale

A data.frame() with one row per dive, per whale

Examples

```
Examples will go here
Examples will go here
```

download_bathy *Download bathymetry data, from FB google drive*

Description

Download bathymetry data, from FB google drive

Usage

```
download_bathy(email, bathy_path = getwd(), overwrite = TRUE)
```

Arguments

email	Email address (for FB Google Drive authentication). You may also be asked to sign in or verify your Google identity as this function runs.
bathy_path	directory in which to save the zip file. Defaults to the current working directory.
overwrite	Whether or not to overwrite existing zip file. Default: TRUE

Value

A zip file with bathymetry data will be downloaded. To use, you'll then need to unzip it.

Examples

```
#examples not yet provided, sorry :(
```

download_drive_acoustic_events *Download spreadsheets with information about acoustic events from the FB Google Drive*

Description

Default is to download data related to all Ziphius SMRT tag datasets (or you can specify the tag(s) for which you want to download data). Data are downloaded from the FB project shared Google Drive.

Usage

```
download_drive_acoustic_events(
  tag_id = zc_smrt_tag_list,
  ae_path = getwd(),
  email,
  overwrite = TRUE,
  type = c("AllClicks", "Buzzes", "PostProcessedEvents")
)
```

Arguments

tag_id	Character string or vector with tag IDs to download (without "-cal.nc" and without additional file path information). Default: all SMRT ziphius tags.
ae_path	Directory (quoted string) where spreadsheet files should be stored locally. Can be one string, or a list the same length as tag_ids. Default: current working directory. Note: use "/" and not "\" to avoid headaches. Final "/" not necessary.
email	Email address (for FB Google Drive authentication). You may also be asked to sign in or verify your Google identity as this function runs.
overwrite	Logical; whether to overwrite existing files. Default is TRUE.
type	The types of acoustic events files to find and download. Options are one or more of: 'AllClicks', 'Buzzes', 'PostProcessedEvents'. Default is all three.

Value

Returns a "dribble" with information about the files and their location on google drive. (The function also downloads the requested files, of course.)

Examples

```
download_drive_nc(email = "sld33@calvin.edu") # (this only works if you know sld33's password...)
```

---

download_drive_nc *Download .nc files with tag data from the FB Google Drive*

Description

Default is to download all Ziphius SMRT tag datasets (or you can specify the tag(s) for which you want to download data). Data are downloaded from the FB project shared Google Drive.

Usage

```
download_drive_nc(
  tag_id = zc_smrt_tag_list,
  path = getwd(),
  email,
  overwrite = TRUE
)
```

Arguments

tag_id	Character string or vector with tag IDs to download (without "-cal.nc" and without additional file path information). Default: all SMRT ziphius tags.
path	Directory (quoted string) where .nc files should be stored. Can be one string, or a list the same length as tag_ids. Default: current working directory. Note: use "/" and not "\" to avoid headaches. Final "/" not necessary.
email	Email address (for FB Google Drive authentication). You may also be asked to sign in or verify your Google identity as this function runs.
overwrite	Logical; whether to overwrite existing files. Default is TRUE.

Value

Returns a "dribble" with information about the files and their location on google drive. (The function also downloads the requested files, of course.)

Examples

```
download_drive_nc(email = "sld33@calvin.edu") # (this only works if you know sld33's password...)
```

`download_drive_rls` *Download raw RL data from the FB Google Drive*

Description

This function downloads the "raw" format of RL data, for all types of anthropogenic sounds, in all available variations of RL type and frequency band. Most users will want to also use `extract_rls` to generate data frames or text files with data in a simpler format.

Usage

```
download_drive_rls(
  rl_file = "RLs_3obank",
  ping_log_file = "qPing_log_corr_times_master",
  path = getwd(),
  email,
  overwrite = TRUE
)
```

Arguments

`rl_file`	name of .csv file (on Google Drive) with ping log data. Defaults to: RLs_3obank.csv (most users should not need to change this, and at present there IS no other file.)
`ping_log_file`	name of .csv file (on Google Drive) with ping log data. Defaults to: qPing_log_corr_times_master.csv (most users should not need to change this, and at present there IS no other file.)
`path`	Quoted string with the path to the directory where you want to save the downloaded file. Defaults to the current working directory. Final "/" not needed. Use "/" rather than "\" to avoid possible headaches.
`email`	Email address (for FB Google Drive authentication). You may also be asked to sign in or verify your Google identity as this function runs.
`overwrite`	Whether or not to overwrite an existing file. Logical. Default: TRUE.

Value

Returns a "dribble" with information about the files and their location on google drive. (The function also downloads the requested files, of course.)

Examples

```
download_drive_nc(email = "sld33@calvin.edu") # (this only works if you know sld33's password...)
```

`extract_rls` *extract RL data*

Description

This function processes the "raw" format of RL data, allowing users to obtain RLs of desired types and in selected frequency bands. If you need direct access to the (more complicated) raw data, see `download_drive_rls`.

This function pulls data on start times (and other metadata) about acoustic wav files recorded by SMRT tags.

Usage

```
extract_rls(
  rl_file,
  ping_log_file,
  email,
  save_output = TRUE,
  path = getwd(),
  out_file,
  signal = "MFAS",
  overwrite = TRUE
)

extract_rls(
  rl_file,
  ping_log_file,
  email,
  save_output = TRUE,
  path = getwd(),
  out_file,
  signal = "MFAS",
  overwrite = TRUE
)
```

Arguments

`rl_file`	name (with path, if needed) of locally-stored .csv file with raw RL data. If not provided, the default is to use `download_drive_rls` to obtain it from the FB Google Drive.
`ping_log_file`	name (with path, if needed) of locally-stored .csv file with raw RL data. If not provided, the default is to use `download_drive_rls` to obtain it from the FB Google Drive.
`email`	Email address (required for FB Google Drive authentication; optional if `rl_file` is provided). You may also be asked to sign in or verify your Google identity as this function runs.
`save_output`	Logical; whether or not to save results in a .csv file. Default is TRUE.
`path`	Quoted string with the path to the directory where you want to save the output file. Defaults to the current working directory. Final "/" not needed. Use "/" rather than "\" to avoid possible headaches.

out_file	Name (quoted string) for output .csv file with results. Optional; default file name is constructed based on signal type and frequency band requested.
signal	Quoted string (or vector of them) indicating which signal types to return RLs for. Options are one or more of: 'MFAS', 'Echosounder', 'Explosive'. Default: 'MFAS'. Note: CAS is currently marked as MFAS type. You can isolate CAS pings and events by getting MFAS RLs and then keeping only pings with duration 20 seconds or longer.
overwrite	Whether or not to overwrite an existing output file. Logical. Default: TRUE.

Details

For MFAS: Events are marked type "MFA". This also includes CAS exposures (which, if we have separated them out before, it is by choosing MFA pings of duration more than 20 seconds). Measurements were made in ANSI-standard 1/3 octave bands centered from 1-40kHz. For MFA events, bands with center frequencies less than 9kHz are considered. For max RMS level, Units are: dB re 1 muPa. Measured in 200 msec windows; reported level is the highest in any one window. (Following conventions of SOCAL BRS, 3S projects.) For Echosounder events, selected bands are between 10-14kHz. For SPLs in individual 1/3 octave bands, the RL is "missing" (NA) if the SNR in that band was less than 6dB (and/or if the frequency band in question is no relevant for the selected signal type). For Explosions, all 1/3 octave bands below 5 kHz are included.

For MFAS: Events are marked type "MFA". This also includes CAS exposures (which, if we have separated them out before, it is by choosing MFA pings of duration more than 20 seconds). Measurements were made in ANSI-standard 1/3 octave bands centered from 1-40kHz. For MFA events, bands with center frequencies less than 9kHz are considered. For max RMS level, Units are: dB re 1 muPa. Measured in 200 msec windows; reported level is the highest in any one window. (Following conventions of SOCAL BRS, 3S projects.) For Echosounder events, selected bands are between 10-14kHz. For SPLs in individual 1/3 octave bands, the RL is "missing" (NA) if the SNR in that band was less than 6dB (and/or if the frequency band in question is no relevant for the selected signal type). For Explosions, all 1/3 octave bands below 5 kHz are included.

Value

Returns a data.frame with RLs (one row per ping). If save_output is true, also saves a csv file in directory path with filename out_file.csv with the results.

Returns a data.frame with RLs (one row per ping). If save_output is true, also saves a csv file in directory path with filename out_file.csv with the results.

Examples

```
extract_rls(email = "sld33@calvin.edu") # (this only works if you know sld33's password...)
extract_rls(email = "sld33@calvin.edu") # (this only works if you know sld33's password...)
```

---

interval_join *Join data at different time-scales*

Description

Given one dataset where each row represents an interval at coarser scale (for example, one row per dive with dive start/end times), pull in summary information about finer-time-scale data. For example, add the max RL during each dive, or the median MSA during each dive.

Usage

```
interval_join(
  x,
  y,
  start_x,
  start_y,
  end_x = start_x,
  end_y = NULL,
  suffix = c("", ".new"),
  ...,
  keep = FALSE
)
```

Arguments

x	interval data frame or tibble (data with one row per dive, dive-cycle, day, etc.) Should have columns that provide start and end times of each interval; these can be date-times or any numeric time indicator
y	data frame at finer time-scale with information to summarize and pull into x
start_x	name of variable with interval start times in x
start_y	name of variable in y that contains sample times or event-start times
end_x	name of variable in x that contains interval end times (defaults to start_x if not given)
end_y	name of variable in y that contains event or sample end times (defaults to start_y if not given)
suffix	(Passed to left_join; most users can ignore) If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2.
...	Additional arguments to pass to left_join
keep	(Passed to left_join; most users can ignore and keep the default value, FALSE) Should the join keys from both x and y be preserved in the output?

Value

A data.frame like the input interval dataset x, but with additional columns for the new summarized variables

Examples

```
Examples will go here
```

---

solar_elev *Add time columns to time-series*

Description

Add time (in seconds since tagon and perhaps as datetimes in UTC and/or local timezone) to a tibble or data-frame with regularly sampled data

Usage

```
solar_elev(time, lat, long, time_zone = "America/Los_Angeles")
```

Arguments

lat	latitude
long	longitude
time_zone	Time zone for local time; defaults to 'America/Los_Angeles'
utc_time	UTC time

Value

solar stage (categorical): "Day", "Night", "Dusk" or "Dawn"

Examples

```
Examples will go here
```

---

solar_stage *Add time columns to time-series*

Description

Add time (in seconds since tagon and perhaps as datetimes in UTC and/or local timezone) to a tibble or data-frame with regularly sampled data

Usage

```
solar_stage(time, lat, long, time_zone = "America/Los_Angeles")
```

Arguments

lat	latitude
long	longitude
time_zone	Time zone for local time; defaults to 'America/Los_Angeles'
utc_time	UTC time

Value

solar stage (categorical): "Day", "Night", "Dusk" or "Dawn"

Examples

```
Examples will go here
```

sum_rls *Sum RLs in dB*

Description

Add up RLs in several frequency bands. Used by `extract_rls`.

Usage

```
sum_rls(rls, energy = FALSE)
```

Arguments

`rls`	numeric vector with RLs in dB
`energy`	Whether or not the RLs are in energy units (alternative is intensity)

Value

sum of the received levels in `rls`

Examples

```
coming soon
```

utc_to_local *Add local time column to data frame with UTC time*

Description

Given an input data frame that already has UTC datetimes, add another datetime column giving the local time

Usage

```
utc_to_local(x, utc_var, tz = "America/Los_Angeles")
```

Arguments

`x`	input data must contain a column that gives the UTC times to be converted to local
`utc_var`	name of variable (in `x`) that gives the UTC times
`tz`	desired time zone name – for example the default, 'America/Los_Angeles'

Value

data.frame `x` with one added column called local_time

Examples

```
Examples will go here
```

Index