Package 'PAMpal'

December 17, 2020

Type Package

Title Load and Process Passive Acoustic Data

Version 0.9.14

Maintainer Taiki Sakai <taiki.sakai@noaa.gov>

Description Tools for loading and processing passive acoustic data. Read in data that has been processed in 'Pamguard' (<https://www.pamguard.org/>), apply a suite processing functions, and export data for reports or external modeling tools. Parameter calculations implement methods by Oswald et al (2007) <doi:10.1121/1.2743157>, Griffiths et al (2020) <doi:10.1121/10.0001229> and Baumann-Pickering et al (2010) <doi:10.1121/1.3479549>.

License GNU General Public License

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), PamBinaries (>= 1.3.0), dplyr, tuneR

Imports seewave, gam, data.table, PAMmisc, RSQLite, stringr, purrr, methods, magrittr, signal, readr, tidyr, ggplot2, manipulate, knitr, rstudioapi, xml2, rjson, rlang, reticulate, lubridate

RoxygenNote 7.1.1

Suggests testthat

NeedsCompilation no

Author Taiki Sakai [aut, cre],
Jay Barlow [ctb],
Emily Griffiths [ctb],
Michael Oswald [ctb],
Simone Baumann-Pickering [ctb],
Julie Oswald [ctb]

Repository CRAN

Date/Publication 2020-12-17 10:10:06 UTC

R topics documented:

AcousticEvent-class . . . 3
AcousticStudy-class . . . 3
addBinaries . . . 4
addCalibration . . . 5
addDatabase . . . 6
addFunction . . . 7
addGps . . . 8
addRecordings . . . 9
calculateAverageSpectra . . . 10
calculateICI . . . 12
calculateModuleData . . . 13
checkStudy . . . 14
export_banter . . . 15
exStudy . . . 16
filter.AcousticStudy . . . 17
getBinaryData . . . 18
getDetectorData . . . 19
is.AcousticEvent . . . 20
is.AcousticStudy . . . 20
is.PAMpalSettings . . . 21
matchEnvData,AcousticEvent-method . . . 21
PAMpal.accessors . . . 23
PAMpalSettings . . . 27
PAMpalSettings-class . . . 28
plotDataExplorer . . . 29
plotWaveform . . . 30
processPgDetections . . . 31
removeBinaries . . . 32
removeCalibration . . . 33
removeDatabase . . . 34
removeFunction . . . 35
roccaWhistleCalcs . . . 36
setSpecies . . . 36
standardCepstrumCalcs . . . 38
standardClickCalcs . . . 38
testCeps . . . 39
testClick . . . 40
testWhistle . . . 41
updateFiles . . . 41
writeEventClips . . . 42
writeWignerData . . . 43

Index **45**

AcousticEvent-class AcousticEvent *Class*

Description

An S4 class storing acoustic detections from an Acoustic Event as well as other related metadata

Slots

`id` unique id or name for this event

`detectors` a list of data frames that have acoustic detections and any measurements calculated on those detections. Each data frame is named by the detector that made the detection

`localizations` a named list storing localizations, named by method

`settings` a list of recorder settings

`species` a list of species classifications for this event, named by classification method (ie. BANTER model, visual ID)

`files` a list of files used to create this object, named by the type of file (ie. binaries, database)

`ancillary` a list of miscellaneous extra stuff. Store whatever you want here

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

AcousticStudy-class AcousticStudy *Class*

Description

An S4 class storing acoustic data from an entire AcousticStudy

Slots

`id` a unique id for the study

`events` a list of AcousticEvent objects with detections from the AcousticStudy

`files` a list of folders and files containing the AcousticStudy data

`gps` a data frame of gps coordinates for the entire AcousticStudy

`pps` the PAMpalSettings object used to create this object

`settings` a named list of various settings for detectors, localizers, etc.

`effort` something about effort lol

`models` a place to store any models run on your data

`ancillary` miscellaneous extra data

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

addBinaries *Add Binaries to a PAMpalSettings Object*

Description

Adds more binary files to the "binaries" slot of a PAMpalSettings object. Interactively asks user to supply folder location if not provided.

Usage

```
addBinaries(pps, folder = NULL, verbose = TRUE)
```

Arguments

pps	a PAMpalSettings object to add binary files to
folder	a folder of binaries to add, all subfolders will also be added
verbose	logical flag to show messages

Value

the same PAMpalSettings object as pps, with the binary files contained in folder added to the "binaries" slot. Only binary files for Click Detector and WhistlesMoans modules will be added, since these are the only types PAMpal currently knows how to process

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# not recommended to create PPS like this, for example only
pps <- new('PAMpalSettings')
binFolder <- system.file('extdata', 'Binaries', package='PAMpal')
pps <- addBinaries(pps, binFolder)
pps
```

addCalibration *Add a Calibration File to a PAMpalSettings Object*

Description

Adds a new calibration function to the "calibration" slot of a PAMpalSettings object. Interactively asks user to supply file and other parameters if not supplied.

Usage

```
addCalibration(
  pps,
  calFile = NULL,
  module = "ClickDetector",
  calName = NULL,
  all = FALSE,
  units = NULL
)

applyCalibration(pps, module = "ClickDetector", all = FALSE)
```

Arguments

pps	a PAMpalSettings object to add a database to
calFile	a calibration file name. Must be csv format with two columns. The first column must be the frequency (in Hz), and the second column must be the sensitivity (in dB), and the columns should be labeled `Frequency` and `Sensitivity`. Can also be supplied as a dataframe in which case the `calName` argument should also be set
module	the Pamguard module type this calibration should be applied to, for now this is only for ClickDetector modules. This is left as an option for future-proofing purposes but should not be needed.
calName	the name to assign to the calibration function, defaults to the file name and only needs to be set if supplying a dataframe instead of a csv file
all	logical flag whether or not to apply calibration to all functions without asking individually, recommended to stay as `FALSE`
units	a number from 1 to 3 specifying the units of the calibration file, number corresponds to dB re V/uPa, uPa/Counts, or uPa/FullScale respectively. A NULL (default) or other value will prompt user to select units.

Details

When adding a calibration, you will be asked what units your calibration value is in. The wave clips stored by Pamguard are values from -1 to 1, so if your calibration is expecting different units then this needs to be accounted for in order to get an accurate SPL value. For V / uPa you must know the voltage range of your recording equipment, and for calibrations expecting Count data you must

know the bit rate of your recordings. If your calibration is already relative to full-scale, then nothing needs to be adjusted. If you don't know the units of your calibration and you are only interested in relative dB levels, then you can select the Full-Scale options.

The calibration function created takes frequency (in Hz) as input and outputs the associated dB value that needs to be added to correct the power spectrum of a signal. If the input is a matrix or dataframe, the first column is assumed to be frequency.

Value

the same PAMpalSettings object as pps, with the calibration function added to the `calibration` slot.

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
pps <- new('PAMpalSettings')
calFile <- system.file('extdata', 'calibration.csv', package='PAMpal')
pps <- addCalibration(pps, calFile, all = TRUE, units=3)
calClick <- function(data, calibration=NULL) {
    standardClickCalcs(data, calibration=calibration, filterfrom_khz = 0)
}
pps <- addFunction(pps, calClick, module = 'ClickDetector')
pps <- applyCalibration(pps, all=TRUE)
pps
```

---

addDatabase *Add a Database to a PAMpalSettings Object*

Description

Adds a new function to the "function" slot in a PAMpalSettings object. Interactively asks for database files if none are supplied as input

Usage

```
addDatabase(pps, db = NULL, verbose = TRUE)
```

Arguments

pps	a PAMpalSettings object to add a database to
db	a database to add
verbose	logical flag to show messages

Value

the same PAMpalSettings object as pps, with the database db added to the "db" slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# not recommended to create a pps like this, for example only
pps <- new('PAMpalSettings')
db <- system.file('extdata', 'Example.sqlite3', package='PAMpal')
pps <- addDatabase(pps, db)
pps
```

---

addFunction *Add a Function to a PAMpalSettings Object*

Description

Adds a new function to the "function" slot in a PAMpalSettings object. Must be run interactively, user will be prompted to assign values for any parameters in the function to be added

Usage

```
addFunction(pps, fun, module = NULL, verbose = TRUE)
```

Arguments

pps	a PAMpalSettings object to add a function to
fun	function to add OR another PAMpalSettings object. In this case all functions from the second object will be added to pps
module	Pamguard module output this function should act on, one of ClickDetector, WhistlesMoans, or Cepstrum. If NULL (default), user will be prompted to select which module it applies to
verbose	logical flag to show messages

Value

the same PAMpalSettings object as pps, with the function fun added to the "functions" slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# not recommended to create a pps like this, for example only
pps <- new('PAMpalSettings')
if(interactive()) pps <- addFunction(pps, standardClickCalcs)
pps <- addFunction(pps, roccaWhistleCalcs, module='WhistlesMoans')
```

---

addGps *Add GPS Locations to an AcousticStudy*

Description

Add GPS Lat / Long to an AcousticStudy or AcousticEvent. If GPS data is not present in any of the databases, user will interactively be asked to provide GPS data to add

Usage

```
addGps(x, gps = NULL, thresh = 3600, ...)

## S4 method for signature 'data.frame'
addGps(x, gps = NULL, thresh = 3600, ...)

## S4 method for signature 'AcousticEvent'
addGps(x, gps = NULL, thresh = 3600, ...)

## S4 method for signature 'list'
addGps(x, gps = NULL, thresh = 3600, ...)

## S4 method for signature 'AcousticStudy'
addGps(x, gps = NULL, thresh = 3600, ...)

## S4 method for signature 'ANY'
addGps(x, gps = NULL, thresh = 3600, ...)
```

Arguments

x	data to add GPS coordinates to. Must have a column UTC, and can also have an optional column Channel
gps	a data frame of GPS coordinates to match to data from x. Must have columns UTC, Latitude, Longitude, and optionally Channel. If not provided and x is an AcousticEvent or AcousticStudy object, then the gps data will be read from the databases contained in the files slot of x
thresh	maximum time apart in seconds for matching GPS coordinates to data, if the closest coordinate is more than thresh apart then the Latitude and Longitude values will be set to NA
...	additional arguments for other methods

Details

Latitude and Longitude coordinates will be matched to the data by using data.tables rolling join with `roll='nearest'`. After the join is done, the time difference between the matched rows is checked and any that are greater than the set threshold are set to NA. This is done to prevent accidentally matching weird things if an incomplete set of GPS data is provided.

If `x` is an AcousticEvent or AcousticStudy, then `gps` can be omitted and will be read from the databases contained in the `files` slot of `x`. If `x` is an AcousticStudy, then the gps data will also be saved to the `gps` slot of the object, and an additional argument `bounds` can be provided. This is a length two vector of `POSIXct` class times that will bound the times of gps data to store, gps data outside this range will not be stored (to reduce the potentially very large amount of data stored in the `gps` slot)

Value

the same data as `x`, with Lat/Long data added. AcousticStudy objects will have all GPS data used added to the "gps" slot, and all AcousticEvents will have Latitude and Longitude added to all detector dataframes

Author(s)

Taiki Sakai `<taiki.sakai@noaa.gov>`

Examples

```
data(exStudy)
# need to update database file to local directory
db <- system.file('extdata', 'Example.sqlite3', package='PAMpal')
exStudy <- updateFiles(exStudy, db=db, bin=NA, verbose=FALSE)
exStudy <- addGps(exStudy)
head(gps(exStudy))
```

---

addRecordings *Add Recordings to an AcousticStudy Object*

Description

Adds recording files to an AcousticStudy object, runs interactively to allow users to select files if they are not provided. No actual recordings are stored, a dataframe containing information on the start and end times of the recording files is added to the object.

Usage

```
addRecordings(x, folder = NULL, log = NULL, progress = TRUE)
```

Arguments

x	a AcousticStudy object to add recordings to
folder	a folder of recordings to add. If NULL, user will be prompted to select a folder of recordings for each database present in x. If a single folder, this will be applied to all databases. If multiple folders, length must be equal to the number of databases and they will be applied to each database in the provided order.
log	(optional) log files for SoundTrap recordings. These are used to adjust apparent lengths of recordings for missing data. If NULL, user will be prompted to provide a folder (selecting no folder is a valid option here). If FALSE this step will be skipped. If a single folder or multiple folders will be applied similar to folder
progress	logical flag to show progress bars

Value

the same object as x with recording information added to the files slots. The information added is a dataframe containing the start and end times of recording

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
recs <- system.file('extdata', 'Recordings', package='PAMpal')
exStudy <- addRecordings(exStudy, folder=recs, log=FALSE, progress=FALSE)
files(exStudy)$recordings
```

---

calculateAverageSpectra

Calculate Average Spectra of Clicks

Description

Calculates the average spectra of all the clicks present in an event

Usage

```
calculateAverageSpectra(
  x,
  evNum = 1,
  calibration = NULL,
  wl = 1024,
  filterfrom_khz = 0,
  filterto_khz = NULL,
```

```
  sr = NULL,
  norm = TRUE,
  plot = TRUE
)
```

Arguments

x	an AcousticEvent or AcousticStudy object
evNum	if x is a study, the event number to calculate the average spectra for
calibration	a calibration function to apply, if desired
wl	the size of the click clips to use for calculating the spectrum. If greater than the clip present in the binary, clip will be zero padded
filterfrom_khz	frequency in khz of highpass filter to apply, or the lower bound of a bandpass filter if filterto_khz is not NULL
filterto_khz	if a bandpass filter is desired, set this as the upper bound. If only a highpass filter is desired, leave as the default NULL value. Currently only highpass and bandpass filters are supported, so if filterfrom_khz is left as zero then this parameter will have no effect
sr	a sample rate to use if the sample rate present in the database needs to be overridden (typically only needed if a decimator was used)
norm	logical flag to normalize magnitudes to 0-1 range
plot	logical flag whether or not to plot the result. The plot will be a two panel plot, the top is a concatenated spectrogram where the y-axis is frequency and the x-axis is click number. The bottom plot is the average spectrogram of all clicks, the y-axis is normalized magnitude (dB values for each click are normalized between 0 and 1 before averaging), x-axis is frequency.

Value

invisibly returns a list with three items: freq, the frequency, average, the average spectra of the event, and all, the individual spectrum of each click in the event as a matrix.

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
# need to update binary file locations to users PAMpal installation
binUpd <- system.file('extdata', 'Binaries', package='PAMpal')
dbUpd <- system.file('extdata', package='PAMpal')
exStudy <- updateFiles(exStudy, bin = binUpd, db=dbUpd)
avSpec <- calculateAverageSpectra(exStudy)
str(avSpec$average)
range(avSpec$freq)
str(avSpec$all)
```

`calculateICI` *Calculate Inter-Click Interval*

Description

Calculate inter-click interval for click data

Usage

```
calculateICI(
  x,
  time = c("UTC", "peakTime"),
  callType = c("click", "whistle", "cepstrum"),
  ...
)

## S4 method for signature 'AcousticStudy'
calculateICI(
  x,
  time = c("UTC", "peakTime"),
  callType = c("click", "whistle", "cepstrum"),
  ...
)

## S4 method for signature 'AcousticEvent'
calculateICI(
  x,
  time = c("UTC", "peakTime"),
  callType = c("click", "whistle", "cepstrum"),
  ...
)

getICI(x, type = c("value", "data"))
```

Arguments

`x`	a AcousticStudy object, a list of AcousticEvent objects, or a single AcousticEvent object
`time`	the time measurement to use. `start` will use the `UTC` value, `peak` will use the `peakTime` value if present (currently present in `standardClickCalcs`, this is the time of the peak of the waveform)
`callType`	the call type to calculate ICI for, usually this is `click` but also allows users to specify `whistle` or `cepstrum` to calculate this using other detector data
`...`	not currently used
`type`	the type of data to return, one of 'value' or 'data'. 'value' returns the single ICI value for each detector, 'data' returns all the individual ICI values used to calculate the number returned by 'value'

Details

Calculates the ICI for each individual detector and across all detectors. ICI calculation is done by ordering all individual detections by time, then taking the difference between consecutive detections and approximating the mode value.

Value

the same object as x, with ICI data added to the "ancillary" slot of each AcousticEvent. Two items will be added. $ici contains all of the individual inter-click intervals used to calculate the ICI, as well as an "All" ICI using all the combined data. $measures will also have a ICI measurement added for each detector, this will be the single modal value. Data in the $measures spot can be exported easily to modeling algorithms. `getICI` will just return either the values stored in $ici for `type = 'data'` or the values stored in $measures for `type = 'value'`

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# setting up example data
data(exStudy)
exStudy <- calculateICI(exStudy)
# each event has its ICI data stored separately, these are 0
# because there is only a single click in this event
ancillary(exStudy[[1]])$ici
# also saves it in measures that will get exported for modeling
ancillary(exStudy[[1]])$measures
```

---

calculateModuleData *Run Custom Calculations on Pamguard Module Data*

Description

Run a list of custom calculations on a Pamguard binary file.

Usage

```
calculateModuleData(
  binData,
  binFuns = list(ClickDetector = list(standardClickCalcs))
)
```

Arguments

binData	Pamguard binary data as read in by loadPamguardBinaryFile
binFuns	A named list of functions to run on each Pamguard module. Currently supported modules are 'ClickDetector' and 'WhistlesMoans', a sample input for binFuns would be list('ClickDetector'=list(clickFun1, clickFun2), 'WhistlesMoans'=list(wmFun1))

Value

A data frame with one row for each channel of each detection. Each row will have the UID, channel number, and name of the detector. Clicks of different classifications are treated as different detectors for this purpose, with the classification label number appended to the detector name. The number of columns will depend on the results of the calculations from the supplied binFuns.

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

---

checkStudy *Check an AcousticStudy Object for Issues*

Description

Checks for any possible issues in an AcousticStudy object, issuing warnings and saving the messages

Usage

```
checkStudy(x, maxLength = Inf, maxSep = 60 * 60 * 2)
```

Arguments

x	an AcousticStudy object
maxLength	events with length greater than this value in seconds will trigger a warning
maxSep	events containing consecutive detections greater than maxSep seconds apart will trigger a warning. This is used to check for situations where detections were possibly added to the incorrect event.

Details

This function is called at the end of processPgDetections with default parameters, but can also be called later to investigate issues specific to each user's data. For example, if you are expecting to process data where all recordings were duty cycled to record 2 out of every 10 minutes, then setting maxLength = 60*2 will alert you to any events that are longer than the 2 minute duty cycle. For continuously recorded data, the maxSep argument can be used to identify situations where there are large gaps between detections in a single event, since this could mean that detections were accidentally added to the incorrect event number during processing.

Value

returns a list of warning messages

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)

# checks if any peak frequencies are 0, so we'll force this
exStudy[[1]][[1]]$peak <- 0
checkStudy(exStudy)
checkStudy(exStudy, maxLength = 1, maxSep = 1)
```

---

export_banter *Export Data for a BANTER Model*

Description

Exports data from an AcousticStudy into the format required to run a BANTER model from the "banter" package

Usage

```
export_banter(
  x,
  dropVars = NULL,
  dropSpecies = NULL,
  training = TRUE,
  verbose = TRUE
)
```

Arguments

x	a AcousticStudy object or a list of AcousticEvent objects
dropVars	a vector of the names of any variables to remove
dropSpecies	a vector of the names of any species to exclude
training	logical flag whether or not this will be used as a training data set, or a value between 0 and 1 specifying what percent of the data should be used for training (with the rest set aside for testing). If TRUE or greater than 0, must contain species ID. NOTE: if value is not 0, 1, TRUE, or FALSE, output will be further split into training and test items within the list output
verbose	logical flag to show summary and informational messages

Value

a list with three items, events, detectors, and na. If value of training is not 0, 1, TRUE, or FALSE, output will be split into training and test lists that contain events and detectors. events is a dataframe with two columns. event.id is a unique identifier for each event, taken from the names of the event list. species is the species classification, taken from the species slot labelled id. detectors is a list of data frames containing all the detections and measurements. There is one list for each unique detector type found in the detectors slots of x. The data frames will only have columns with class numeric, integer, factor, or logical, and will also have columns named UID, Id, parentUID, sampleRate, Channel, angle, and angleError, removed so that these are not treated as parameters for the banter random forest model. The dataframes will also have columns event.id and call.id added. na contains the UIDs and Binary File names for any detections that had NA values. These cannot be used in the random forest model and are removed from the exported dataset.

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# setting up example data
data(exStudy)
exStudy <- setSpecies(exStudy, method='pamguard')
banterData <- export_banter(exStudy)
# drop some variables
banterLess <- export_banter(exStudy, dropVars = c('peak', 'duration'))
```

---

exStudy *Example AcousticStudy Object*

Description

An example AcousticStudy object created using the example PAMpalSettings object provided with the package. Processed with mode='db'

Usage

```
data(exStudy)
```

Format

a AcousticStudy object containing two AcousticEvent objects

`filter.AcousticStudy` *Filter an AcousticStudy or AcousticEvent Object*

Description

Apply dplyr-like filtering to the detecitons of an AcousticStudy or AcousticEvent object, with a special case for filtering by species for an AcousticStudy

Usage

```
## S3 method for class 'AcousticStudy'
filter(.data, ..., .preserve = FALSE)
```

Arguments

`.data`	AcousticStudy or AcousticEvent to filter
`...`	Logical expressions, syntax is identical to filter. There is a special case if `.data` is an AcousticStudy object where a logical expression using `species` or `Species` will filter by the species present in the `$id` of the `species` slot within each AcousticEvent
`.preserve`	not used

Value

The original `.data` object, filtered by the given logical expressions

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# create example data
data(exStudy)
exStudy <- setSpecies(exStudy, method='manual', value=letters[1:2])
filterData <- filter(exStudy, peak < 20)
getDetectorData(filterData)$click

filterData <- filter(exStudy, species == 'a')
species(filterData[[1]])
```

getBinaryData *Get Raw Binary Data for Detections*

Description

Fetches matching binary data from a single or multiple detections in an AcousticEvent object

Usage

```
getBinaryData(x, UID, quiet = FALSE, ...)
```

Arguments

x	a AcousticStudy object, a list of AcousticEvent objects, or a single AcousticEvent object
UID	the UID(s) of the individual detections to fetch the binary data for
quiet	logical flag to quiet some warnings, used internally and should generally not be changed from default FALSE
...	additional arguments to pass to loadPamguardBinaryFile

Value

a list of PamBinary objects for each UID

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
binData <- getBinaryData(exStudy, UID = 8000003)
# works with multiple UIDs, if UIDs arent present they will be ignored
binData <- getBinaryData(exStudy, UID = c(8000003, 529000024, 1))
```

getDetectorData *Extract and Combine Detector Data*

Description

Extracts just the detector data from all of x, and will combine all detections from each call type (currently whistle, click, and cepstrum) into a single data frame.

Usage

```
getDetectorData(x)

getClickData(x)

getWhistleData(x)

getCepstrumData(x)
```

Arguments

x	data to extract detector data from, either an AcousticStudy, AcousticEvent or list of AcousticEvent object

Details

The purpose of this function is to extract your data out of PAMpal's S4 classes and put them into an easier format to work with. The output will be a list of up to three data frames, one for each call type found in your data. Each different call type will have had different processing applied to it by processPgDetections. Additionally, each detector will have its associated event id, the name of the detector, and the species id attached to it (species will be NA if not set). All detections from each call type will be combined into a single large data frame

Value

A list of data frames containing all detection data from x, named by call type ('click', 'whistle', or 'cepstrum').

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
dets <- getDetectorData(exStudy)
names(dets)
str(dets$click)
```

```
# works on single events as well
oneDets <- getDetectorData(exStudy[[1]])
str(oneDets$click)
```

`is.AcousticEvent` *Check if an Object is an AcousticEvent*

Description

Function to check if an object is an AcousticEvent

Usage

```
is.AcousticEvent(x)
```

Arguments

`x`	object to check

`is.AcousticStudy` *Check if an Object is an AcousticStudy*

Description

Function to check if an object is an AcousticStudy

Usage

```
is.AcousticStudy(x)
```

Arguments

`x`	object to check

is.PAMpalSettings *Check if an Object is a PAMpalSettings*

Description

Function to check if an object is a PAMpalSettings

Usage

```
is.PAMpalSettings(x)
```

Arguments

x	object to check

matchEnvData,AcousticEvent-method *Match Environmental Data to an AcousticStudy Object*

Description

Extracts all variables from a netcdf file matching Longitude, Latitude, and UTC coordinates of the start of each AcousticEvent object. Matched values are stored in the "ancillary" slot of each event

Usage

```
## S4 method for signature 'AcousticEvent'
matchEnvData(
  data,
  nc = NULL,
  var = NULL,
  buffer = c(0, 0, 0),
  FUN = c(mean, median, sd),
  fileName = NULL,
  progress = TRUE,
  ...
)

## S4 method for signature 'AcousticStudy'
matchEnvData(
  data,
  nc = NULL,
  var = NULL,
  buffer = c(0, 0, 0),
  FUN = c(mean, median, sd),
```

```
  fileName = NULL,
  progress = TRUE,
  ...
)
```

Arguments

data	an AcousticStudy or AcousticEvent object that must have GPS data added to it using the addGps functions
nc	name of a netcdf file, ERDDAP dataset id, or an edinfo object
var	(optional) vector of variable names
buffer	vector of Longitude, Latitude, and Time (seconds) to buffer around each datapoint. All values within the buffer will be used to report the mean, median, and standard deviation
FUN	a vector or list of functions to apply to the data. Default is to apply mean, median, and standard deviation calculations
fileName	(optional) file name to save downloaded nc file to. If not provided, then no nc files will be stored, instead small temporary files will be downloaded and then deleted. This can be much faster, but means that the data will need to be downloaded again in the future. If `fileName` is provided, then the function will attempt to download a single nc file covering the entire range of your data. If your data spans a large amount of time and space this can be problematic.
progress	logical flag to show progress bar
...	other parameters to pass to ncToData

Value

original data object with environmental data added to the `ancillary` slot of each event. Complete data will be stored in `ancillary(data)$environmental`, and the mean of each downloaded variable will be stored in `ancillary(data)$measures` so that it can be exported for modeling. For each event the coordinates associated with the earliest UTC value in that event are used to match

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
nc <- system.file('extdata', 'sst.nc', package='PAMmisc')
# suppressing warnings because nc coordinates dont align with this data,
# function warns of possible coordinate mismatch
exStudy <- suppressWarnings(matchEnvData(exStudy, nc=nc, progress=FALSE))
str(ancillary(exStudy[[1]])$environmental)
ancillary(exStudy[[1]])$measures
```

PAMpal.accessors AcousticEvent *and* AcousticStudy *accessors*

Description

Accessors for slots in AcousticEvent and AcousticStudy objects

Usage

```
settings(x, ...)

## S4 method for signature 'AcousticEvent'
settings(x, ...)

settings(x) <- value

## S4 replacement method for signature 'AcousticEvent'
settings(x) <- value

localizations(x, ...)

## S4 method for signature 'AcousticEvent'
localizations(x, ...)

localizations(x) <- value

## S4 replacement method for signature 'AcousticEvent'
localizations(x) <- value

id(x, ...)

## S4 method for signature 'AcousticEvent'
id(x, ...)

id(x) <- value

## S4 replacement method for signature 'AcousticEvent'
id(x) <- value

detectors(x, ...)

## S4 method for signature 'AcousticEvent'
detectors(x, ...)

detectors(x) <- value

## S4 replacement method for signature 'AcousticEvent'
```

```
detectors(x) <- value

species(x, ...)

## S4 method for signature 'AcousticEvent'
species(x, ...)

## S4 method for signature 'AcousticStudy'
species(x, type = "id", ...)

species(x) <- value

## S4 replacement method for signature 'AcousticEvent'
species(x) <- value

files(x, ...)

## S4 method for signature 'AcousticEvent'
files(x, ...)

files(x) <- value

## S4 replacement method for signature 'AcousticEvent'
files(x) <- value

ancillary(x, ...)

## S4 method for signature 'AcousticEvent'
ancillary(x, ...)

ancillary(x) <- value

## S4 replacement method for signature 'AcousticEvent'
ancillary(x) <- value

## S4 method for signature 'AcousticEvent,ANY,ANY,ANY'
x[i]

## S4 replacement method for signature 'AcousticEvent,ANY,ANY,ANY'
x[i] <- value

## S4 method for signature 'AcousticEvent'
x$name

## S4 replacement method for signature 'AcousticEvent'
x$name <- value

## S4 method for signature 'AcousticEvent,ANY,ANY'
```

```
x[[i]]

## S4 replacement method for signature 'AcousticEvent,ANY,ANY,ANY'
x[[i]] <- value

## S4 method for signature 'AcousticStudy'
id(x, ...)

## S4 replacement method for signature 'AcousticStudy'
id(x) <- value

## S4 method for signature 'AcousticStudy'
files(x, ...)

## S4 replacement method for signature 'AcousticStudy'
files(x) <- value

gps(x, ...)

## S4 method for signature 'AcousticStudy'
gps(x, ...)

gps(x) <- value

## S4 replacement method for signature 'AcousticStudy'
gps(x) <- value

## S4 method for signature 'AcousticStudy'
detectors(x, ...)

events(x, ...)

## S4 method for signature 'AcousticStudy'
events(x, ...)

events(x) <- value

## S4 replacement method for signature 'AcousticStudy'
events(x) <- value

## S4 method for signature 'AcousticStudy'
settings(x, ...)

## S4 replacement method for signature 'AcousticStudy'
settings(x) <- value

effort(x, ...)
```

```
## S4 method for signature 'AcousticStudy'
effort(x, ...)

effort(x) <- value

## S4 replacement method for signature 'AcousticStudy'
effort(x) <- value

pps(x, ...)

## S4 method for signature 'AcousticStudy'
pps(x, ...)

pps(x) <- value

## S4 replacement method for signature 'AcousticStudy'
pps(x) <- value

## S4 method for signature 'AcousticStudy'
ancillary(x, ...)

## S4 replacement method for signature 'AcousticStudy'
ancillary(x) <- value

models(x, ...)

## S4 method for signature 'AcousticStudy'
models(x, ...)

models(x) <- value

## S4 replacement method for signature 'AcousticStudy'
models(x) <- value

## S4 method for signature 'AcousticStudy,ANY,ANY,ANY'
x[i]

## S4 replacement method for signature 'AcousticStudy,ANY,ANY,ANY'
x[i] <- value

## S4 method for signature 'AcousticStudy'
x$name

## S4 replacement method for signature 'AcousticStudy'
x$name <- value

## S4 method for signature 'AcousticStudy,ANY,ANY'
x[[i]]
```

```
## S4 replacement method for signature 'AcousticStudy,ANY,ANY,ANY'
x[[i]] <- value
```

Arguments

x	a AcousticEvent or AcousticStudy object
...	other arguments to pass to methods
value	value to assign with accessor
type	species type to select
i	index of the object to access
name	name of the object to access

Value

id a unique id or name for this object

settings a named list of settings for each detector and localization or recorder

detectors a list of detector data frames

localizations list of localizations

species list of species classifications

files list of files used to create this object

events a list of AcousticEvent objects

gps a dataframe containing gps data

pps the PAMpalSettings object used to create this

effort something about effort?

ancillary miscellaneous extra data

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

PAMpalSettings *Constructor for PAMpalSettings Object*

Description

Create a PAMpalSettings object. Any values that are not supplied will be asked for interactively. Three processing functions will also be added by default: standardClickCalcs, roccaWhistleCalcs, and standardCepstrumCalcs

Usage

```
PAMpalSettings(db = NULL, binaries = NULL, verbose = TRUE)
```

Arguments

db	the full path to a Pamguard database file
binaries	a folder containing Pamguard binary files, all subfolders will also be added
verbose	logical flag to show messages

Value

A PAMpalSettings object

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# can be run with no arguments with popup menu selections
if(interactive()) pps <- PAMpalSettings()
db <- system.file('extdata', 'Example.sqlite3', package='PAMpal')
bin <- system.file('extdata', 'Binaries', package='PAMpal')
# or data folders can be supplied ahead of time
if(interactive()) pps <- PAMpalSettings(db=db, binaries=bin)
```

PAMpalSettings-class PAMpalSettings *Class*

Description

An S4 class that stores settings related to all processing and analysis steps done in PAMpal. A PAMpalSettings object will be the main input to any major function in the PAMpal package.

Slots

`db` the full path to a PamGuard database file

`binaries` a list with items "folder" containing the directory of the PamGuard binary files, and "list" containing the full path to each individual binary file.

`functions` a named list of functions to apply to data read in by PAMpal. Should be named by the PamGuard module the function should be applied to. Currently supports "ClickDetector", "WhistlesMoans", and "Cepstrum".

`calibration` a named list of calibration functions to apply while applying functions from the "functions" slot. Should named by the PamGuard module, same as the "functions"

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

plotDataExplorer *Explore Data in an Interactive Plot*

Description

Creates an interactive plot of detector data. Allows user to choose which numeric data to plot, and will allow user to both color and facet the plot by any columns that are characters or factors

Usage

```
plotDataExplorer(x, callType = NULL, maxCategories = 15)
```

Arguments

x	data to plot, can be an AcousticStudy, AcousticEvent, data.frame or a list of AcousticEvent objects
callType	the specific type of call to plot. If NULL (default), will prompt user to choose which type if more than one is present.
maxCategories	maximum number of categories to color and facet by. Only character and factor data with a number of unique values less than or equal to this number will be shown as options for selecting colors and facets. Not recommended to increase this value much beyond 20, trying to plot a large number of colors will cause R to be sad.

Value

nothing, just plots

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)

if(interactive()) plotDataExplorer(exStudy)
if(interactive()) plotDataExplorer(exStudy, callType='click')
```

plotWaveform *Plot Graphical Representations of Waveforms*

Description

Fetches matching binary data from a single or multiple detections in an AcousticStudy object, then plot the resulting data

Usage

```
plotWaveform(x, UID)

plotSpectrogram(x, UID, sr = NULL, ...)

plotWigner(x, UID, sr = NULL, ...)
```

Arguments

x	a AcousticStudy object, a list of AcousticEvent objects, or a single AcousticEvent object
UID	the UID(s) of the individual detections to fetch the binary data for
sr	if NULL (default) will try to read sample rate from your data. If provided as a value will override sample rate in the data.
...	other arguments to pass to the spectrogram or wigner functions

Details

The plotSpectrogram function uses the function specgram to plot the spectrogram, see this function for plotting options. The plotWigner function uses the function wignerTransform to plot the Wigner-Ville transform, see this function for options.

Value

Nothing, just shows plots for every channel of the waveform for each UID provided

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
plotWaveform(exStudy, 8000003)
plotSpectrogram(exStudy, 8000003)
plotWigner(exStudy, 8000003)
```

processPgDetections *Load and Process Detections from Pamguard*

Description

Loads and processes acoustic detection data that has been run through Pamguard. Uses the binary files and database(s) contained in pps, and will group your data into events by the grouping present in the 'OfflineEvents' and 'Detection Group Localiser' tables (mode = 'db') or by the grouping specified by start and end times in the supplied grouping (mode = 'time'). Will apply all processing functions in pps to the appropriate modules

Usage

```
processPgDetections(
  pps,
  mode = c("db", "time"),
  id = NULL,
  grouping = NULL,
  format = "%Y-%m-%d %H:%M:%OS",
  progress = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

pps	a PAMpalSettings object containing the databases, binaries, and functions to use for processing data. See PAMpalSettings. Can also be an AcousticStudy object, in which case the pps slot will be used.
mode	selector for how to organize your data in to events. db will organize by events based on tables in the databases, and time will organize into events based on timestamps provided in grouping.
id	an event name or id for this study, will default to today's date if not supplied (recommended to supply your own informative id)
grouping	For mode = 'db', the table to group events by. Either event to use the OfflineEvents table, or detGroup to use the detection group localiser module groups. For mode = 'time', this should be a data frame with three mandatory columns and 1 row for each separate event. The mandatory columns are start, end, and id. start and end should specify the start and end time of the event and must be in UTC. id should specify a unique id for each event. There are also optional columns species, db, and sr. species should provide a species ID if it is available. db and sr are the corresponding database and sample rate to associate with a particular event, these typically do not need to be specified as the function will attempt to automatically match them based on the times of the events and the databases. Note that db must be the full filepath to the database.

	If a clear match is not found then the user will be prompted to either select from a list or input the proper sample rate. grouping can be supplied either as a data frame or as a filepath to a csv file.
format	the date format for the start and end columns in grouping if it is a csv. Times are assumed to be UTC. See details section of strptime for more information on how to properly format this
progress	logical flog to show progress bars
verbose	logical flag to show messages
...	additional arguments to pass onto to different methods

Value

an AcousticStudy object with one AcousticEvent for each event in the events slot, and the PAMpalSettings object used stored in the pps slot.

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
exPps <- new('PAMpalSettings')
exPps <- addDatabase(exPps, system.file('extdata', 'Example.sqlite3', package='PAMpal'))
exPps <- addBinaries(exPps, system.file('extdata', 'Binaries', package='PAMpal'))
exClick <- function(data) {
    standardClickCalcs(data, calibration=NULL, filterfrom_khz = 0)
}
exPps <- addFunction(exPps, exClick, module = 'ClickDetector')
exPps <- addFunction(exPps, roccaWhistleCalcs, module='WhistlesMoans')
exPps <- addFunction(exPps, standardCepstrumCalcs, module = 'Cepstrum')
# process events labelled within the Pamguard database
exStudyDb <- processPgDetections(exPps, mode='db', id='Example')
# can also give an AcousticStudy as input and it will use same functions and data
reprocess <- processPgDetections(exStudyDb, mode='db', id='Reprocess')
# process events with manually set start/end times
grp <- data.frame(start = as.POSIXct('2018-03-20 15:25:10', tz='UTC'),
                  end = as.POSIXct('2018-03-20 15:25:11', tz='UTC'),
                  id = 'GroupExample')
exStudyTime <- processPgDetections(exPps, mode='time', grouping=grp, id='Time')
```

---

removeBinaries *Remove Binaries from a PAMpalSettings Object*

Description

Remove a binary folder and associated files from the "binaries" slot in a PAMpalSettings object.

Usage

```
removeBinaries(pps, index = NULL)
```

Arguments

pps	a PAMpalSettings object to remove binaries from
index	index indicating which binary folders to remove. Can be a vector if you want to remove multiple folders. If missing user is prompted to select a folder from a list, will only show up to the first 20. You can easily remove all of the folders with a large index like `1:1000`

Value

the same PAMpalSettings object as pps, with the binary folders and files associated with those folders removed from the "binaries" slot.

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
exPps <- new('PAMpalSettings')
exPps <- addBinaries(exPps, system.file('extdata', 'Binaries', package='PAMpal'))
removeBinaries(exPps,index = 1)
if(interactive()) removeBinaries(exPps)
```

---

removeCalibration *Remove a Calibration Function from a PAMpalSettings Object*

Description

Remove a calibration function from the "calibration" slot of a PAMpalSettings object

Usage

```
removeCalibration(pps, index = NULL, module = "ClickDetector", verbose = TRUE)
```

Arguments

pps	a PAMpalSettings object to remove a calibration from
index	index of the calibration function to remove. If `NULL`, user will be prompted to select from a list. This can also be a vector to remove multiple calibration functions at once.
module	the module of the calibration function to remove, currently not needed
verbose	logical flag to show messages

Value

the same PAMpalSettings object as pps, with the calibration function removed from the "calibration" slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
pps <- new('PAMpalSettings')
calFile <- system.file('extdata', 'calibration.csv', package='PAMpal')
pps <- addCalibration(pps, calFile, all = TRUE, units=3)
calClick <- function(data, calibration=NULL) {
    standardClickCalcs(data, calibration=calibration, filterfrom_khz = 0)
}
pps <- addFunction(pps, calClick, module = 'ClickDetector')
pps <- applyCalibration(pps, all=TRUE)
pps
removeCalibration(pps, index=1)
```

---

removeDatabase *Remove a Database from a PAMpalSettings Object*

Description

Remove a database from the "db" slot in a PAMpalSettings object.

Usage

```
removeDatabase(pps, index = NULL)
```

Arguments

pps	a PAMpalSettings object to remove a database from
index	index indicating which database(s) to remove. Can be a vector if you want to remove multiple databases. If missing user is prompted to select a database from a list, will only show up to the first 20. You can easily remove all of the databases with a large index like `1:1000`

Value

the same PAMpalSettings object as pps, with the database(s) removed from the "db" slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
exPps <- new('PAMpalSettings')
exPps <- addDatabase(exPps, system.file('extdata', 'Example.sqlite3', package='PAMpal'))
removeDatabase(exPps, 1)
if(interactive()) removeDatabase(exPps)
```

---

removeFunction *Remove a Function from a PAMpalSettings Object*

Description

Remove a function from the "function" slot in a PAMpalSettings object.

Usage

```
removeFunction(pps, index = NULL)
```

Arguments

pps	a PAMpalSettings object to remove a function from
index	index indicating which function to move, counting from ClickDetector functions first, then WhistlesMoans functions, then Cepstrum functions. This is the same order functions appear in when examining the pps object. For example, if there are two Click functions and one Whistle function, the Whistle function would have an index of 3. If missing, user can select from a list. This can also be a vector to remove multiple functions at once.

Value

the same PAMpalSettings object as pps, with the function removed from the "functions" slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
exPps <- new('PAMpalSettings')
exPps <- addFunction(exPps, roccaWhistleCalcs, module='WhistlesMoans')
exPps <- addFunction(exPps, standardCepstrumCalcs, module = 'Cepstrum')
removeFunction(exPps, 1)
removeFunction(exPps, 1:2)
# normally best to use interactively instead of specifying index
if(interactive()) removeFunction(exPps)
```

roccaWhistleCalcs *Calculate a Set of Measurements for Whistles*

Description

Calculate a set of measurements from a whistle contour. All calculations following ROCCA method from Julie and Michael Oswald, as implemented in Pamguard and detailed in Oswald et al (2007) <doi:10.1121/1.2743157>

Usage

```
roccaWhistleCalcs(data)
```

Arguments

data	a list that must have freq the whistle contour stored as a vector of FFT bin frequencies in hertz, and time the time in seconds at each bin.

Value

A list with 50 calculated ROCCA parameters, each item in the list will only have 1 entry so that this can easily be converted to a data frame.

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(testWhistle)
roccaWhistleCalcs(testWhistle)
```

setSpecies *Set the Species Classification of Events*

Description

Sets the species slot of AcousticEvent objects within an AcousticStudy

Usage

```
setSpecies(x, method = c("pamguard", "manual", "reassign"), value, type = "id")
```

Arguments

x	a AcousticStudy object, a list of AcousticEvent objects, or a single AcousticEvent object
method	the method for assigning species to an event. Currently supports pamguard, which will use the 'eventType' or 'Text_Annotation' column to assign species, manual which will use value to assign species manually, or reassign which will use value to reassign an old species label to a new one
value	required only if method is set to 'manual' or 'reassign'. For 'manual', can either be a single value to assign to all events, or a vector with length equal to the number of events. Can also be a dataframe with columns event and species, in which case species will be matched to corresponding event names instead of just relying on the order. If using this, please note the prefix OE or DGL present on most event numbers (see the id slot of your events, or names(events(x))). For 'reassign', value must be a data frame with columns old and new. Any events with species id in the old column of the dataframe will get reassigned to the corresponding id in the new column.
type	the type of classification to set, this is just a label within the species slot. Default 'id' should typically not be changed since this is used by other functions

Value

the same object as x, with species identifications assigned as an item named type in the species slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# setting up example data
exPps <- new('PAMpalSettings')
exPps <- addDatabase(exPps, system.file('extdata', 'Example.sqlite3', package='PAMpal'))
exPps <- addBinaries(exPps, system.file('extdata', 'Binaries', package='PAMpal'))
exClick <- function(data) {
    standardClickCalcs(data, calibration=NULL, filterfrom_khz = 0)
}
exPps <- addFunction(exPps, exClick, module = 'ClickDetector')
exPps <- addFunction(exPps, roccaWhistleCalcs, module='WhistlesMoans')
exPps <- addFunction(exPps, standardCepstrumCalcs, module = 'Cepstrum')
exData <- processPgDetections(exPps, mode='db')
exData <- setSpecies(exData, method='pamguard')
species(exData)
exData <- setSpecies(exData, method='manual', value = c('sp1', 'sp2'))
species(exData)
exData <- setSpecies(exData, method='reassign',
                     value = data.frame(old='sp1', new='sp3'))
species(exData)
```

standardCepstrumCalcs *Calculate a Set of Measurements from a Cepstrum Contour*

Description

Calculate a set of measurements from a cepstrum contour. This is currently used to measure the inter-click interval of the burst pulse type calls

Usage

```
standardCepstrumCalcs(data)
```

Arguments

data	a list that must have quefrency the "quefrency" at each cepstrum contour, sr the sample rate of the data, and time the time in seconds at each bin

Value

A list with inter-click interval (ici), duration (seconds), and slope of the ici

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(testCeps)
standardCepstrumCalcs(testCeps)
```

standardClickCalcs *Calculate a Set of Measurements for Clicks*

Description

Calculate a set of "standard" measurements for odontocete clicks

Usage

```
standardClickCalcs(
  data,
  sr_hz = "auto",
  calibration = NULL,
  filterfrom_khz = 10,
  filterto_khz = NULL,
  winLen_sec = 0.0025
)
```

Arguments

data	a list that must have 'wave' containing the wave form as a matrix with a separate column for each channel, and 'sr' the sample rate of the data. Data can also be a Wave class object, like one created by Wave.
sr_hz	either 'auto' (default) or the numeric value of the sample rate in hertz. If 'auto', the sample rate will be read from the 'sr' of data
calibration	a calibration function to apply to the spectrum, must be a gam. If NULL no calibration will be applied (not recommended).
filterfrom_khz	frequency in khz of highpass filter to apply, or the lower bound of a bandpass filter if filterto_khz is not NULL
filterto_khz	if a bandpass filter is desired, set this as the upper bound. If only a highpass filter is desired, leave as the default NULL value
winLen_sec	length in seconds of fft window. The click wave is first shortened to this number of samples around the peak of the wave, removing a lot of the noise around the click. Following approach of JB/EG/MS.

Details

Calculations of parameters mostly follow the approach outlined in Griffiths et al (2020) <doi:10.1121/10.0001229> and Baumann-Pickering et al (2010) <doi:10.1121/1.3479549>. Additionally, up to 3 highest peak frequencies and the "troughs" between them are calculated (see peakTrough)

Value

A data frame with one row for each channel of click waveform

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(testClick)
standardClickCalcs(testClick)
```

---

testCeps *A fake cepstrum contour*

Description

A manually created fake cepstrum contour, mimicing what the output would be from the Pamguard module and fed into the cepstrum calcs

Usage

```
data(testCeps)
```

Format

A list with three items:

quefrency a vector of the cepstrum contour bin numbers, not actually quefrency

time a vector of the time values of the cepsturm contour in seconds

sr the sample rate of the recording

---

testClick *A two-channel recording of a delphinid click*

Description

An example delphinid click waveform. This is a two-channel recording of some kind of delphinid click, recorded at 192kHz. There are 800 samples recorded on each channel.

Usage

```
data(testClick)
```

Format

A list with two items:

wave a matrix with two columns of 800 samples, each column is a separate recording channel

sr the sample rate of the recording

Source

Southwest Fisheries Science Center / NMFS / NOAA

testWhistle *A fake whistle contour*

Description

A manually created fake whistle contour reanging from 1kHz to 3.1kHz

Usage

```
data(testWhistle)
```

Format

A list with two items:

freq a vector of the frequency contour values in hertz

time a vector of the time values of the contour in seconds

updateFiles *Update Location of Files in an AcousticStudy*

Description

Updates the stored locations of binary, database, and/or recording files in the `files` slots of an AcousticStudy and all AcousticEvent objects within. Runs interactively to prompt users to select folders if missing files are found. Typically used after changing computers, or if original data was on an external hard drive. If any missing files are not able to be located, they will be kept in the files slot so that this function can be run again

Usage

```
updateFiles(x, bin = NULL, db = NULL, recording = NULL, verbose = TRUE)
```

Arguments

`x`	an AcousticStudy or AcousticEvent object
`bin`	folder containing updated binary file locations. If `NULL` (default), user will be prompted to select a folder. If `NA`, binary files will be skipped.
`db`	single file or folder containing updated database file locations. `NULL` (default), user will be prompted to select a folder. If `NA`, database files will be skipped.
`recording`	folder containing updated recording file locations. If `NULL` (default), user will be prompted to select a folder. If `NA`, recording files will be skipped.
`verbose`	logical flag to print messages about success of replacement

Value

the same AcousticStudy and AcousticEvent object as x with updated file locations

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
# files in exStudy will have paths local to package creator's computer
files(exStudy)$db
file.exists(files(exStudy)$db)
files(exStudy)$binaries
file.exists(files(exStudy)$binaries)
# folder with example DB
db <- system.file('extdata', package='PAMpal')
# folder with example binaries
bin <- system.file('extdata', 'Binaries', package='PAMpal')
exStudy <- updateFiles(exStudy, db=db, bin=bin)
files(exStudy)$db
file.exists(files(exStudy)$db)
files(exStudy)$binaries
file.exists(files(exStudy)$binaries)
```

---

writeEventClips *Create Wav Clips of Data*

Description

Creates audio clips containing sounds from events or detections

Usage

```
writeEventClips(
  x,
  buffer = c(-0.1, 0.1),
  outDir = ".",
  mode = c("event", "detection"),
  channel = 1,
  progress = TRUE,
  verbose = TRUE
)
```

Arguments

x	AcousticStudy object containing data to make wav clips for
buffer	amount before and after each event to also include in the clip, in seconds. Can either be a vector of length two specifying how much to buffer before and after (first number should be negative), or a single value if the buffer amount should be identical.
outDir	directory to write clips to, defaults to current directory
mode	either 'event' or 'detection' specifying whether to create wav clips of entire events or individual detections
channel	channel(s) of clips to write
progress	logical flag to show progress bar
verbose	logical flag to show summary messages

Value

A vector of file names for the wav clips that were successfully created, any that were not able to be written will be NA. Note that currently this can only write clips with up to 2 channels. File names will be formatted as [Event or Detection]_[EventId]CH[ChannelNumber(s)]_[YYYYMMDDHHMMSS]_[mmm].wav (the last numbers are the start time of the file in UTC, accurate to milliseconds)

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
recs <- system.file('extdata', 'Recordings', package='PAMpal')
exStudy <- addRecordings(exStudy, folder=recs, log=FALSE, progress=FALSE)
## Not run:
# not running so that no wav clips are written to disk
wavs <- writeEventClips(exStudy, outDir='WavFolder', mode='event')

## End(Not run)
```

---

writeWignerData *Write Wigner Transform Data of Click Detections to Disk*

Description

Create Wigner-Ville transform data of click clips from all detections and save them to disk. A CSV file will also be written that lists all UIDs contained in the output

Usage

```
writeWignerData(
  x,
  n = 256,
  t = 300,
  outDir = ".",
  mode = "nparray",
  progress = TRUE,
  ...
)
```

Arguments

x	AcousticStudy object containing data to make Wigner data for
n	number of frequency bins for Wigner transform (recommended power of 2)
t	number of samples to use for the click clip passed to the transform
outDir	directory to write data to
mode	specifies the kind of output that will be created, currently only supports creating NumPy arrays using the `reticulate` package, in future will support image creation
progress	logical flag to show progress bar
...	optional arguments to pass

Value

A list with two items: `files` - a vector of file names for the Wigner data that were successfully created, any that were not able to be written will be NA, and `warnings`, a list with items containing event IDs that triggered any warnings

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
exStudy <- setSpecies(exStudy, method='pamguard')
## Not run:
# not running because files are written to disk
wigFiles <- writeWignerData(exStudy, outDir = 'WigFolder')

## End(Not run)
```

Index

∗ **datasets**
exStudy, 16
testCeps, 39
testClick, 40
testWhistle, 41
[,AcousticEvent,ANY,ANY,ANY-method *(PAMpal.accessors)*, 23
[,AcousticStudy,ANY,ANY,ANY-method *(PAMpal.accessors)*, 23
[<-,AcousticEvent,ANY,ANY,ANY-method *(PAMpal.accessors)*, 23
[<-,AcousticStudy,ANY,ANY,ANY-method *(PAMpal.accessors)*, 23
[[,AcousticEvent,ANY,ANY-method *(PAMpal.accessors)*, 23
[[,AcousticStudy,ANY,ANY-method *(PAMpal.accessors)*, 23
[[<-,AcousticEvent,ANY,ANY,ANY-method *(PAMpal.accessors)*, 23
[[<-,AcousticStudy,ANY,ANY,ANY-method *(PAMpal.accessors)*, 23
$,AcousticEvent-method *(PAMpal.accessors)*, 23
$,AcousticStudy-method *(PAMpal.accessors)*, 23
$<-,AcousticEvent-method *(PAMpal.accessors)*, 23
$<-,AcousticStudy-method *(PAMpal.accessors)*, 23

AcousticEvent, *3*, *8, 9*, *11, 12*, *15–18*, *22, 23*, *27*, *30*, *32*, *36, 37*, *41, 42*
AcousticEvent-class, 3
AcousticStudy, *8–12*, *14–18*, *22, 23*, *27*, *30–32*, *36, 37*, *41–44*
AcousticStudy-class, 3
addBinaries, 4
addCalibration, 5
addDatabase, 6
addFunction, 7
addGps, 8, *22*
addGps,AcousticEvent-method *(addGps)*, 8
addGps,AcousticStudy-method *(addGps)*, 8
addGps,ANY-method *(addGps)*, 8
addGps,data.frame-method *(addGps)*, 8
addGps,list-method *(addGps)*, 8
addRecordings, 9
ancillary *(PAMpal.accessors)*, 23
ancillary,AcousticEvent-method *(PAMpal.accessors)*, 23
ancillary,AcousticStudy-method *(PAMpal.accessors)*, 23
ancillary<- *(PAMpal.accessors)*, 23
ancillary<-,AcousticEvent-method *(PAMpal.accessors)*, 23
ancillary<-,AcousticStudy-method *(PAMpal.accessors)*, 23
applyCalibration *(addCalibration)*, 5

calculateAverageSpectra, 10
calculateICI, 12
calculateICI,AcousticEvent-method *(calculateICI)*, 12
calculateICI,AcousticStudy-method *(calculateICI)*, 12
calculateModuleData, 13
checkStudy, 14

detectors *(PAMpal.accessors)*, 23
detectors,AcousticEvent-method *(PAMpal.accessors)*, 23
detectors,AcousticStudy-method *(PAMpal.accessors)*, 23
detectors<- *(PAMpal.accessors)*, 23
detectors<-,AcousticEvent-method *(PAMpal.accessors)*, 23

effort *(PAMpal.accessors)*, 23
effort,AcousticStudy-method *(PAMpal.accessors)*, 23

effort<- (PAMpal.accessors), 23
effort<-,AcousticStudy-method
(PAMpal.accessors), 23
events (PAMpal.accessors), 23
events,AcousticStudy-method
(PAMpal.accessors), 23
events<- (PAMpal.accessors), 23
events<-,AcousticStudy-method
(PAMpal.accessors), 23
export_banter, 15
exStudy, 16

files (PAMpal.accessors), 23
files,AcousticEvent-method
(PAMpal.accessors), 23
files,AcousticStudy-method
(PAMpal.accessors), 23
files<- (PAMpal.accessors), 23
files<-,AcousticEvent-method
(PAMpal.accessors), 23
files<-,AcousticStudy-method
(PAMpal.accessors), 23
filter, *17*
filter.AcousticStudy, 17

getBinaryData, 18
getCepstrumData (getDetectorData), 19
getClickData (getDetectorData), 19
getDetectorData, 19
getICI (calculateICI), 12
getWhistleData (getDetectorData), 19
gps (PAMpal.accessors), 23
gps,AcousticStudy-method
(PAMpal.accessors), 23
gps<- (PAMpal.accessors), 23
gps<-,AcousticStudy-method
(PAMpal.accessors), 23

id (PAMpal.accessors), 23
id,AcousticEvent-method
(PAMpal.accessors), 23
id,AcousticStudy-method
(PAMpal.accessors), 23
id<- (PAMpal.accessors), 23
id<-,AcousticEvent-method
(PAMpal.accessors), 23
id<-,AcousticStudy-method
(PAMpal.accessors), 23
is.AcousticEvent, 20
is.AcousticStudy, 20
is.PAMpalSettings, 21

loadPamguardBinaryFile, *14*, *18*
localizations (PAMpal.accessors), 23
localizations,AcousticEvent-method
(PAMpal.accessors), 23
localizations<- (PAMpal.accessors), 23
localizations<-,AcousticEvent-method
(PAMpal.accessors), 23

matchEnvData,AcousticEvent-method, 21
matchEnvData,AcousticStudy-method
(matchEnvData,AcousticEvent-method),
21
models (PAMpal.accessors), 23
models,AcousticStudy-method
(PAMpal.accessors), 23
models<- (PAMpal.accessors), 23
models<-,AcousticStudy-method
(PAMpal.accessors), 23

ncToData, *22*

PAMpal.accessors, 23
PAMpalSettings, *3–7*, *27*, 27, *31–35*
PAMpalSettings-class, 28
peakTrough, *39*
plotDataExplorer, 29
plotSpectrogram (plotWaveform), 30
plotWaveform, 30
plotWigner (plotWaveform), 30
pps (PAMpal.accessors), 23
pps,AcousticStudy-method
(PAMpal.accessors), 23
pps<- (PAMpal.accessors), 23
pps<-,AcousticStudy-method
(PAMpal.accessors), 23
processPgDetections, *14*, 31

removeBinaries, 32
removeCalibration, 33
removeDatabase, 34
removeFunction, 35
roccaWhistleCalcs, *27*, 36

setSpecies, 36
settings (PAMpal.accessors), 23
settings,AcousticEvent-method
(PAMpal.accessors), 23

settings,AcousticStudy-method
(PAMpal.accessors), 23
settings<- (PAMpal.accessors), 23
settings<-,AcousticEvent-method
(PAMpal.accessors), 23
settings<-,AcousticStudy-method
(PAMpal.accessors), 23
specgram, *30*
species (PAMpal.accessors), 23
species,AcousticEvent-method
(PAMpal.accessors), 23
species,AcousticStudy-method
(PAMpal.accessors), 23
species<- (PAMpal.accessors), 23
species<-,AcousticEvent-method
(PAMpal.accessors), 23
standardCepstrumCalcs, *27*, 38
standardClickCalcs, *27*, 38
strptime, *32*

testCeps, 39
testClick, 40
testWhistle, 41

updateFiles, 41

Wave, *39*
wignerTransform, *30*
writeEventClips, 42
writeWignerData, 43