Package 'PAMpal'

May 7, 2026

Type Package

Title Load and Process Passive Acoustic Data

Version 1.5.2

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), dplyr (>= 1.1.1)

Imports PamBinaries (>= 1.3.0), PAMmisc (>= 1.12.0), tuneR, seewave, gam, data.table, RSQLite, purrr, methods, signal, tidyr, ggplot2, knitr, xml2, rlang, reticulate, lubridate, geosphere, shiny, future.apply

RoxygenNote 7.3.2

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 2026-02-26 20:30:02 UTC

Contents

`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>

addAnnotation *Add Annotation Data to an AcousticStudy Object*

Description

Adds annotation data to an AcousticStudy object, usually in preparation for exporting to an external source. Most pieces of the annotation form will be filled in by the user when the function is called, but `lat`, `lon`, `time_start`, `time_end`, `freq_low`, `freq_high`, `source_id`, and `annotation_id` will be filled in automatically based on data in each AcousticEvent. Annotations are stored for each event in the `ancillary` slot.

Usage

```
addAnnotation(x, anno, verbose = TRUE)

prepAnnotation(
  x,
  specMap = NULL,
  mode = c("event", "detection"),
  interactive = FALSE,
```

```
  ...
)

getAnnotation(x)

checkAnnotation(x)

export_annomate(x, file = NULL)

matchRecordingUrl(anno, rec)
```

Arguments

x	an AcousticStudy or AcousticEvent object
anno	an annotation dataframe
verbose	logical flag to print messages
specMap	data.frame to map species ids in x to names to be used for the annotation (ex. from 'ZC' to 'Ziphius cavirostris'. Dataframe must have columns old and new
mode	one of 'event' or 'detection' to create annotation for events or detections
interactive	logical flag to fill annotation data interactively (not recommended)
...	additional named arguments to fill in annotation data. If names match a column in the annotation, that value will be used for all events or detections in the annotation
file	file to write a CSV of the annotations to, if NULL (default) then no file will be written
rec	dataframe of recording url information. Must have column recording_url. If clips were created using writeEventClips, then must have column filename containing the wav file names. Other column names will be automatically parsed from there. If wav files are from another source, must contain columns matchId

Value

The same object as x with an $annotation item added the the ancillary slot of each event in x

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`	database(s) to add, or single directory containing databases
`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
```

---

addFPOD *Add FPOD Detector to an AcousticStudy*

Description

Adds data from FPOD detector CSV files to an AcousticStudy object as new detectors of type "fpod"

Usage

```
addFPOD(x, fpod, detectorName = "FPOD")
```

Arguments

x	an AcousticStudy object
fpod	path(s) to CSV files containing FPOD detector output
detectorName	name for the detector, the default 'FPOD' should be fine unless you want to differentiate between multiple FPOD detectors

Details

FPOD detections are added to events based on their times. All detections between the start and end times events. NOTE: most PAMpal functions were designed with only PAMGuard data in mind, there is a chance that adding FPOD detections will cause other advanced functionality to not work.

Behavior is slightly different depending on how the original AcousticStudy was created. For those processed with mode='db', the start and end times for each event are just determined by the times of detections within the event.

For those processed with mode='recording' or mode='time', the start and end times for each event are determined by the start/end times of the recording files or the grouping file provided initially. This means that it is possible that there are events which initially had zero PAMGuard detections that now have FPOD detections. In these cases a new AcousticEvent will be created that only has FPOD detections, these events may not work with a variety of other PAMpal functions.

Value

the same object as x with FPOD detector data added

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

---

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, default = FALSE, ...)
```

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, Cepstrum, or GPLDetector. If NULL (default), user will be prompted to select which module it applies to
verbose	logical flag to show messages
default	logical flag to use default function parameters if present
...	named arguments to pass to function being added

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, thresh = 3600, keepDiff = FALSE, ...)

## 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
keepDiff	logical flag to keep time difference column (between GPS time and data time)

Details

Latitude and Longitude coordinates will be matched to the data by interpolating between points in the provided GPS data. After the interpolating 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. An approximate distance between the interpolated points and the closest known GPS point is provided as a "gpsUncertainty" column (distance in meters).

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))
```

---

addHydrophoneDepth *Add Hydrophone Depth Data to an AcousticStudy*

Description

Add hydrophone depth to an AcousticStudy or AcousticEvent

Usage

```
addHydrophoneDepth(x, depth = NULL, depthCol = NULL, thresh = 60, ...)
```

Arguments

x	an AcousticStudy to add depth data to
depth	a CSV or data frame of depth values to match to data from x. Must have column UTC, and a column containing depth data to be specified by depthCol. If not provided and x is an AcousticEvent or AcousticStudy object, then the depth data will be read from the databases contained in the files slot of x

depthCol	the name of the column containing depth in the dataframe or database. If left as NULL, will search for a single column containing the word "depth" or "Depth"
thresh	maximum time apart in seconds for matching depth to data, if the closest value is more than thresh apart then the depth value will be set to NA
...	additional arguments for other methods

Details

Depth values will be matched to the data by using data.table's 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 depth data is provided.

If x is an AcousticEvent or AcousticStudy, then depth can be omitted and will be read from the databases contained in the files slot of x.

Value

the same data as x, with depth data added. All AcousticEvents will have depth data added to all detector dataframes as column hpDepth

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 <- addHydrophoneDepth(exStudy)
getClickData(exStudy[1])
```

---

addMeasures *Add Measures*

Description

Adds "measures" to an AcousticStudy or AcousticEvent. A "measure" is an event-level variable that will be exported alongside data from that event

Usage

```
addMeasures(x, measures, replace = TRUE)

getMeasures(x)
```

Arguments

x	an AcousticStudy or AcousticEvent object
measures	the measures to add. Can either be a named list, where names match event names of x or a dataframe with column eventId matching the event names of x. If a list, every item within the list must also be named by the variable name. All other data within measures will be added as new measures
replace	logical flag whether or not to replace

Value

object of same class as x with measures added

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
measList <- list('Example.OE1' = list(a=1, b=2),
                 'Example.OE2' = list(a=2, b=3)
                 )
exMeasure <- addMeasures(exStudy, measList)
print(getMeasures(exMeasure))
measDf <- data.frame(eventId = c('Example.OE1', 'Example.OE2'),
                     a=4:5,
                     b=6:7)
exMeasure <- addMeasures(exMeasure, measDf, replace=TRUE)
getMeasures(exMeasure)
```

---

addNote *addNote*

Description

Adds a note to an AcousticEvent or AcousticStudy. Notes can either be accessed with the "getNotes" function, or up to 6 notes will be printed when the object is printed

Usage

```
addNote(x, to = c("study", "event"), evNum = 1, label = NULL, note)

getNotes(x)
```

Arguments

x	An AcousticStudy or AcousticEvent object
to	One of "study" or "event", which object to add the note to
evNum	If x is an AcousticStudy and to is "event", the number or name of the event(s) to add notes to (can be a vector of numbers or names to add the same note to multiple events)
label	(optional) a short header or label for the note. Recommended to set this as a sumamry of the more detailed note
note	the full note message

Value

For addNote, the same data as x, with notes added. For getNotes, a list of all notes present in x

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
exStudy <- addNote(exStudy, to='study', label='Note1',
                   note='My first note for this study')
exStudy <- addNote(exStudy, to='event', evNum=1:2, label='Note2',
                   note='A note for the first two events')
exStudy <- addNote(exStudy[[1]], to='event', label='Note3',
                   note='A second note for the first event')
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 = FALSE,
  fileFormat = NULL,
  dateFormat = NULL,
```

```
  progress = TRUE
)

mapWavFolder(
  folder = NULL,
  log = NULL,
  progress = TRUE,
  fileFormat = NULL,
  dateFormat = NULL
)
```

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
fileFormat	optional file date format regex to extract date from file names See Details for more information.
dateFormat	matching date format for fileFormat in strptime format. See Details for more information.
progress	logical flag to show progress bars

Details

Custom Date Format using fileFormat and dateFormat. This is optional, and only required if the built in options do not work and you get the warning message "Could not convert wav names to time properly for files..."

fileFormat must be a regular expression designed to extract the numbers that represent the date-time from the recording file, and only these numbers. This portion of the regex must be wrapped in parentheses.

dateFormat must be a strptime style date format that matches the datetime text extracted by fileFormat.

Examples: a wav file named "Recording_20101201-230112.wav" (where the numbers are in yyyymmdd-hhmmss) would have `fileFormat=".*_([0-9]{8}\\-[0-9]{6})\\.wav$"` and `dateFormat="%Y%m%d-%H%M%S"`.

A wav file named "01122010230112_CH2.wav" (where the numbers are in ddmmyyyyhhmmss) would have `fileFormat="([0-9]{14})_CH[0-9]{1}\\.wav$"` and `dateFormat="%d%m%Y%H%M%S"`. Both examples are for a file datetime of "2010-12-01 23:01:12".

mapWavFolder returns a dataframe of start and end times

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
```

---

addSettings *Add Settingss to a PAMpalSettings Object*

Description

Adds settings to a PAMpalSettings object, usually from an XML file created by Pamguard's "Export XML Configuration"

Usage

```
addSettings(pps, settings = NULL, type = c("xml", "list"), verbose = TRUE)
```

Arguments

pps	a PAMpalSettings object to add settings to
settings	settings to add, either an XML file or a
type	one of 'xml' or 'list' indicating type of settings to add
verbose	logical flag to show messages

Value

the same PAMpalSettings object as pps, with a new list of settings replacing what was previously in the "settings" slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# not recommended to create PPS like this, for example only
pps <- new('PAMpalSettings')
xmlSettings <- system.file('extdata', 'Example.xml', package='PAMpal')
pps <- addSettings(pps, xmlSettings, type='xml')
```

`addWaveHeight` *Add Wave Height Data to an AcousticStudy*

Description

Add wave height to an AcousticStudy or AcousticEvent

Usage

```
addWaveHeight(x, height, thresh = 3600)
```

Arguments

`x`	an AcousticStudy to add height data to
`height`	either a single numeric value, or a dataframe with column `UTC` and either column `waveHeight` specifying height (m) at that time, or `beaufort` specifying the beaufort sea state at that time
`thresh`	maximum time apart in seconds for matching height to data, if the closest value is more than `thresh` apart then the height value will be set to `NA`

Details

height values will be matched to the data by using data.table's 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 height data is provided.

Value

the same data as `x`, with wave height data added. All AcousticEvents will have height data added to all detector dataframes as column `waveHeight`

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
# need to update database file to local directory
exStudy <- addWaveHeight(exStudy, height=.5)
getClickData(exStudy[1])
```

`bindStudies` *Combine AcousticStudy Objects*

Description

Combines multiple AcousticStudy objects (or lists of these) into a single object

Usage

```
bindStudies(...)
```

Arguments

...	AcousticStudy objects, or a list of AcousticStudy objects

Details

All events will be combined into one large list of events. Files, settings, effort, models, GPS, and ancillary fields will be combined using the squishList function from the PAMmisc package (dataframes are combined, vectors are appended). The id is changed by pasting all IDs together along with a note that they have been combined. Note that the PAMpalSettings object in the pps slot is just left as the `pps` in the first AcousticStudy to be combined, and thus is not representative of the new combined AcousticStudy

Value

A single AcousticStudy object

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

`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 = 512,
  channel = 1:2,
  filterfrom_khz = 0,
  filterto_khz = NULL,
  sr = NULL,
  snr = 0,
  norm = TRUE,
  plot = TRUE,
  noise = FALSE,
  decimate = 1,
  sort = FALSE,
  mode = "spec",
  title = NULL,
  ylim = NULL,
  flim = NULL,
  cmap = hcl.colors(30, "YlOrRd", rev = TRUE),
  brightness = 0,
  contrast = 0,
  q = 0.01,
  showBreaks = TRUE,
  ...
)
```

Arguments

`x`	an AcousticEvent or AcousticStudy object
`evNum`	if `x` is a study, the event index number to calculate the average spectra for. Note that this is the index in the order that they appear in the AcousticStudy object, not the actual event number. Alternatively full event names can be used
`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

channel	channel(s) to include in calculations. Currently does not correspond to actual channel in instrument, just the order present in the binary file
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 not needed)
snr	minimum signal-to-noise ratio to be included in the average, in dB. SNR is calculated as difference between the signal and noise spectra at the peak frequency of the signal. This can be inaccurate if noise is inaccurate (see noise for issues with noise calculations)
norm	logical flag to normalize dB magnitude to maximum of 0
plot	logical flag whether or not to plot the result. This will create two plots, the first is a concatenated spectrogram where the y-axis is frequency and the x-axis is click number. The second plot is the average spectrogram of all clicks, the y-axis is dB, x-axis is frequency. Can be a vector of length two to create only one of the two plots
noise	logical flag to plot an average noise spectrum. This is estimated by taking a window of length wl immediately before click. Since there are only a limited number of samples saved in the Pamguard binary files, this can be very inaccurate when wl is a large proportion of the total samples saved. In these cases the noise floor will appear nearly identical to the signal, reducing wl can help get a more accurate noise floor.
decimate	integer factor to reduce sample rate by
sort	logical flag to sort concatenated spectrogram by peak frequency
mode	one of 'spec' or 'ceps' to plot the spectrum or cepstrum
title	replacement titles for plots, can be length vector of length two to provide separate titles
ylim	optional y limits for mean spectra plot
flim	optional frequency limits for both plots
cmap	colors to use for concatenated click spectrogram, either a palette function or vector of colors
brightness	value from -255 to 255, positive values increase brightness, negative values decrease brightness of concatenated spectrogram image
contrast	value from -255 to 255, positive values increase contrast, negative values decrease contrast of concatenated spectrogram image
q	lower and upper quantiles to remove for scaling concatenated spectrogram. Or if a single value, then quantiles q and 1-q will be used. Ex. if q=.01, then the bottom 1 plotting the image. This is done purely for cosmetic reasons, no output data is affected
showBreaks	logical flag to show lines separating events when plotting multiple events
...	optional args

Value

invisibly returns a list with six items: `freq` - the frequency, `UID` - the UID of each click, `avgSpec` - the average spectra of the event, `allSpec` - the individual spectrum of each click in the event as a matrix with each spectrum in a separate column, `avgNoise` - the average noise spectra, `allNoise` - the individual noise spectrum for each click, `snrVals` - the estimated signal-to-noise ratio for each click

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$avgSpec)
range(avSpec$freq)
str(avSpec$allSpec)
```

---

calculateEchoDepth *Calculate Depth from Echoes*

Description

Calculate the estimated depth of echolocation clicks using surface reflected echoes. This uses the time delay between the received signal and its surface echo to estimate the depth of a calling animal. Requires that a set of waveform clips has been created using writeEventClips, and that events have been localized.

Usage

```
calculateEchoDepth(
  x,
  wav,
  clipLen = 0.03,
  spParams = NULL,
  soundSpeed = 1500,
  hpDepthError = 1,
  locType = "PGTargetMotion",
  plot = TRUE,
  nPlot = 400,
  nCol = 5,
  plotDir = NULL,
```

```
  plotIci = TRUE,
  maxIci = 2.5,
  sr = NULL,
  progress = TRUE,
  verbose = TRUE
)
```

Arguments

x	AcousticStudy object
wav	either folder containing wave clips or list of wave clip files
clipLen	length (seconds) of clip to analyze
spParams	list of species-specific parameters, see details
soundSpeed	sound speed (meters/second) to use for calculations
hpDepthError	maximum error (meters) in hydrophone depth measurement
locType	name of localization, note that this function is not computing any localization, only using previously calculated
plot	logical flag to create summary plots
nPlot	number of waveform plots to create for summary
nCol	number of columns for waveform summary plot
plotDir	directory to store plot outputs, default NULL will result in no plots being created
plotIci	logical flag to additionally create ICI plots
maxIci	maximum allowed ICI value (seconds)
sr	if not NULL (default), clips in wav will be decimated to match this sample rate
progress	logical flag to show progress bar
verbose	logical flag to show messages

Details

spParams allows for species-specific filtering and acceptable echo time delays to be specified. These are provided as a list with elements freqLow and freqHigh specifying the lower and upper ends of a bandpass filter to apply to the signals (in Hz), which can aid in properly detecting the echoes. Parameters minTime and maxTime can also be supplied to define ranges on allowed time delay values. Alternatively if maxTime is NULL or not present it will be calculated from the hydrophone geometry, and minTime can be calculated from geometry by providing minDepth and maxRange as the minimum detectable depth and maximum detectable range (in meters).

If the same values for these parameters should be used for all detections in x, then spParams can be provided as a list with each parameter named, e.g.

```
list(freqLow=10e3, freqHigh=50e3, minTime=.001, maxTime=NULL)
```

If different values should be used for different species, then spParams must be a named list where the names match the species in x, providing a separate list of values for each species. e.g.

```
list(Zc=list(freqLow=10e3, freqHigh=50e3, minTime=.001, maxTime=NULL),
     Pm=list(freqLow=2e3, freqHigh=16e3, minTime=.001, maxTime=NULL))
```

Value

the AcousticStudy object x with estimated dive depth outputs added for each detection that had a matching wav clip file in wav. Detections that either did not have matchinf wav files or did not have localizations will have NA for all dive depth outputs. The depth outputs are

maxTime Delay time with maximum correlation value

pair2Time Delay time with second highest correlation value

pair3Time Delay time with third highest correlation value

maxMag Correlation magnitude for "maxTime"

pair2Mag Correlation magnitude for "pair2Time"

pair3Mag Correlation magnitude for "pair3Time"

maxDepth Calculated depth for "maxTime"

pair2Depth Calculated depth for "pair2Time"

pair3Depth Calculated depth for "pair3Time"

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# example not run because it requires access to large files not present
# in the package testing material
## Not run:
study <- addRecordings(study, folder='path/to/recordings')
wavPath <- 'path/to/wavFiles'
writeEventClips(study, outDir=wavPath, mode='detection')
study <- calculateEchoDepth(study, wav=wavPath)

## End(Not run)
```

---

calculateICI *Calculate Inter-Click Interval*

Description

Calculate inter-click interval for click data

Usage

```
calculateICI(
  x,
  time = c("UTC", "peakTime"),
  iciRange = c(0, Inf),
  callType = c("click", "whistle", "cepstrum", "gpl"),
  verbose = TRUE,
  ...
)

## S4 method for signature 'AcousticStudy'
calculateICI(
  x,
  time = c("UTC", "peakTime"),
  iciRange = c(0, Inf),
  callType = c("click", "whistle", "cepstrum", "gpl"),
  verbose = TRUE,
  ...
)

## S4 method for signature 'AcousticEvent'
calculateICI(
  x,
  time = c("UTC", "peakTime"),
  iciRange = c(0, Inf),
  callType = c("click", "whistle", "cepstrum", "gpl"),
  verbose = TRUE,
  ...
)

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)
iciRange	optional range of allowed ICI (time to next detection) values. Values outside of this range will be removed before calculating the modal ICI.
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
verbose	logical flag to print messages
...	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 $measures for `type = 'value'` or a dataframe of the individual ICI values used to calculate these (with columns indicating separate Channels, eventIds, and detectorNames) for `type = 'data'`

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)),
  settings = NULL
)
```

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))
settings	a list of settings from a Pamguard XML file

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, parentID, 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 are special cases to filter by environmental variables, species ID, database, or detector name. See details.
`.preserve`	not used

Details

Most expression provided will be used to filter out detections based on calculated parameters.

If the name of an environmental variable added using matchEnvData,AcousticStudy-method is provided, will filter to only events with environmental variables matching those conditions.

If a provided logical expression uses `"species"` or `"Species"`, then events will be filtered using the species present in the `$id` of the `species` slot of each event.

If a provided logical expression uses `"database"` or `"Database"`, then only events with databases matching the expression in `files(.data)$db` will remain

If a provided logical expression uses `"detector"` or `"Detector"`, then only detections from detectors with names matching the expression will remain in events. Any events left with no detections will be removed.

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]])
```

`filterEchoDepths` *Filter Candidate Echo Depths*

Description

Filter out possible echo depths from calculateEchoDepth based on maximum depth, autocorrelation magnitude, and maximum swim speed criteria. Requires that calculateEchoDepth has been run first. This function adds a `keepClick` column to the data to track which detections should be used for further depth analysis by marking them as `FALSE` to be excluded or `TRUE` to be used

Usage

```
filterEchoDepths(
  x,
  time = 30,
  depth = NULL,
  speed = NULL,
  maxDepth = 4000,
  minCorr = 0.01
)
```

Arguments

`x`	an AcousticStudy object that has been processed with calculateEchoDepth
`time`	maximum time apart (seconds) for detections. Detections with no no other detection within `time` seconds will be marked as `FALSE`
`depth`	maximum depth difference (meters) between consecutive clicks, this value should be determined by maximum swim speed
`speed`	as an alternative to providing `depth`, the swim speed (meters / second) can be provided and then `depth` will be calculated as `time` * `speed`
`maxDepth`	calculated depth values greater than this will be marked `FALSE`
`minCorr`	detections with autocorrelation magnitude less than this will be marked as `FALSE`

Value

the AcousticStudy x with detections marked with column keepClick as TRUE or FALSE depending if they pass the filter parameters

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# example not run because \link{calculateEchoDepth} must be run first,
# and it requires a large amount of data not stored in the package
## Not run:
study <- calculateEchoDepth(study, wav='path/to/wavFiles')
study <- filterEchoDepths(study, time=30, speed=50/30, maxDepth=4000)

## End(Not run)
```

---

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,
  type = c("click", "whistle", "cepstrum", "gpl"),
  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
type	detection type
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))
```

---

getClipData *Get Wav Clips of Data*

Description

Reads audio clips containing sounds from events or detections

Usage

```
getClipData(
  x,
  buffer = c(0, 0.1),
  mode = c("event", "detection"),
  channel = 1,
  useSample = FALSE,
  fixLength = FALSE,
  fillZeroes = TRUE,
  progress = TRUE,
  verbose = TRUE,
  FUN = NULL,
  ...
)
```

Arguments

x	AcousticStudy object containing data to read 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.
mode	either 'event' or 'detection' specifying whether to create wav clips of entire events or individual detections

channel	channel(s) of clips to write
useSample	logical flag to use startSample information in binaries instead of UTC time for start of detections. This can be slightly more accurate (~1ms) but will take longer
fixLength	logical flag to fix the output clip length to a constant value. If TRUE, then output clip length is entirely determined by the buffer value, as if the detection or event had zero length. E.g. buffer=c(-2,1) will produce clips 3 seconds long, starting 2 seconds before the detection/event start time.
fillZeroes	logical flag to fill gaps in non-consecutive clips with zeroes. If FALSE, will give warnings when attempting to retrieve clips spanning non-consecutive files and return no clip data
progress	logical flag to show progress bar
verbose	logical flag to show summary messages
FUN	optional function to apply to wav clips. This function takes default inputs wav, a Wave class object, name the name of the detection or event, time the start and end time of the clip, channel as above, mode as above, and additional args ...
...	optional arguments to pass to FUN

Value

A named list of wav clips

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 <- getClipData(exStudy, mode='event')

## End(Not run)
```

---

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, cepstrum, and gpl) into a single data frame.

Usage

```
getDetectorData(x, measures = TRUE)

getClickData(x, measures = TRUE)

getWhistleData(x, measures = TRUE)

getCepstrumData(x, measures = TRUE)

getGPLData(x, measures = TRUE)

getFPODData(x, measures = TRUE)

nDetections(x, distinct = FALSE)

nClicks(x, distinct = FALSE)

nWhistles(x)

nCepstrum(x)

nGPL(x)
```

Arguments

x	data to extract detector data from, either an AcousticStudy, AcousticEvent or list of AcousticEvent object
measures	logical flag whether or not to append measures to detector dataframes
distinct	logical flag to only return number of distinct click detections

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', 'cepstrum', or 'gpl').

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)
```

`getWarnings` *Get Warning Messages*

Description

Accessor to easily get all warning messages for `x`

Usage

```
getWarnings(x)
```

Arguments

`x`	an AcousticStudy or AcousticEvent object

Value

a list of warning messages, named by the function call that created the warning

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
## Not run:
data(exStudy)
# This will trigger a warning, then we can access it
exStudy <- filter(exStudy, species == "test")
getWarnings(exStudy)

## End(Not run)
```

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

loadPamguardXML *Load Pamguard XML Settings*

Description

Loads in relevant settings and formats for use in PAMpal

Usage

```
loadPamguardXML(x)
```

Arguments

x	an XML file created by Pamguard's "Export XML Configuration"

Value

A list with settings for audio `sources` (sound acuisition, decimators, FFT, and cepstrum) and `detectors` (click detector and whistle and moan detector). Also stores the entire XML file as `raw` and the file name as `file`

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
xmlFile <- system.file('extdata', 'Example.xml', package='PAMpal')
xmlList <- loadPamguardXML(xmlFile)
str(xmlList)
```

markAnnotated *Mark Detections as Annotated*

Description

Marks detections within an AcousticStudy as being within the bounds of an annotation box. Annotations can either be read in from the "Spectrogram Annotation" module of PAMguard, or supplied as a separate dataframe. Detections must be entirely contained within the annotation bounds.

Usage

```
markAnnotated(
  x,
  anno = NULL,
  tBuffer = 0,
  fBuffer = 0,
  table = "Spectrogram_Annotation"
)
```

Arguments

x	an AcousticStudy object
anno	annotations to read from. If NULL, will be read in from the PAMguard database. If a data.frame, must have columns start and end in UTC, and column id. Can additionally have columns fmin and fmax to apply frequency bounds (values in Hz).
tBuffer	additional buffer value to add on to annotation time bounds in seconds. If a single number, the number of seconds to extend the bounds by on the start and end of each annotation. Can also be a vector of two to extend different values on the start and end. This can be useful if original bounding boxes were drawn very close to the desired detections since any small portion of a signal outside the box will cause it to be excluded.
fBuffer	additional buffer value to add to annotation frequency bounds in Hz. If a single number, the number of Hz to extend bounds by on lower and upper end of boxes. Can also be a vector of two to extend different values on lower and upper bounds. This can be useful if original bounding boxes were drawn very close to the desired detections since any small portion of a signal outside the box will cause it to be excluded.
table	if anno is NULL, the name of the "Spectrogram Annotation" module table within the database.

Details

This adds new columns inAnno and annoId to all detector dataframes within the AcousticStudy. inAnno is a logical flag whether or not a given detection was fully contained in any annotation bounding box, and annoId lists the IDs of the boxes it matched. A detection is considered within an annotation only if it is entirely within the time and frequency bounds of the annotation. For GPL and whistle detections, the min and max frequency values are used. For click detections, only the peak frequency is used. For cesptrum detections, frequency bounds are ignored.

Value

the same object as x, but detectors have additional columns added

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
annotation <- data.frame(start = min(getWhistleData(exStudy)$UTC),
                         fmin = c(16000, 17000),
                         fmax = c(17000, 18000))
annotation$end <- annotation$star + 1
exStudy <- markAnnotated(exStudy, annotation)
getWhistleData(exStudy)[c('UTC', 'duration', 'freqMin', 'freqMax', 'inAnno', 'annoId')]
```

---

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),
  fileName = NULL,
  progress = TRUE,
  depth = 0,
  ...
)

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

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
depth	depth values (meters) to use for matching, overrides any Depth column in the data or can be used to specify desired depth range when not present in data. Variables will be summarised over the range of these depth values. NULL uses all available depth values
...	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
```

`matchTimeData` *Match time-based data to PAMpal objects*

Description

Match any time-based data (dataframe with a `UTC` column) to an AcousticStudy or AcousticEvent object

Usage

```
matchTimeData(
  x,
  data,
  mode = c("event", "detection"),
  thresh = Inf,
  interpolate = TRUE,
  replace = FALSE,
  keepDiff = FALSE
)

timeJoin(x, y, thresh = Inf, interpolate = TRUE, replace = FALSE)
```

Arguments

`x`	AcousticStudy or AcousticEvent object to match data to, or a dataframe for `timeJoin`
`data`	a data frame to match to data to `x`. Must have column `UTC`, and optionally a column `db` if subsets of data should be matched only to parts of `x` with that database. All other columns will be considered variables to add to `x`
`mode`	one of "event" or "detection". "event" will match one set of variables per event, stored in the "measures" for that event. "detection" will match variables to every detection.
`thresh`	maximum time apart in seconds for matching to data, if the closest value is more than `thresh` apart then the variable values will be set to `NA`
`interpolate`	logical flag whether or not to interpolate between points in `data` or just matched to nearest time
`replace`	one of `TRUE`, `FALSE`, or `NA`. If `TRUE`, all existing values with the same name as columns in `data` will be replaced. If `FALSE` no replacement occurs. If `NA` only values which are `NA` will be replaced with new values
`keepDiff`	logical flag to keep time difference data
`y`	dataframe to join to `x`

Details

This function lets you match any arbitrary data to a PAMpal object as long as it has a time associated with it. Data will be attached to detector dataframes for `mode='detection'` or to the event "measures" location for `mode='event'` (this is where calculateICI and matchEnvData,AcousticStudy-method store their event data). These can be accessed with the getMeasures function and are also exported in the various "getXXX" functions (getClickData etc.) if `measures=TRUE` (default).

All columns in the provided `data` object will be treated as variables to add, with a few exceptions. There are a few reserved column names used by PAMpal that cannot be overridden (e.g. UID, eventId, species). Also any columns already existing in the PAMpal data will not be overridden unless `replace` is not `FALSE`. The column names in `data` will be used as the names for the added variables, so care should be taken to ensure these are informative enough for future use.

Value

the same data as `x`, with data added from `data`

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
addData <- data.frame(UTC = as.POSIXct('2018-03-20 15:25:10', tz='UTC'),
                      newVariable = 26)
data <- matchTimeData(exStudy, addData, mode='detection')
getClickData(data)
data <- matchTimeData(exStudy, addData, mode='event')
getMeasures(data)
```

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, j, ...]]

## 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,missing'
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
j	not used

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,
  settings = NULL,
  functions = NULL,
  verbose = TRUE,
  default = FALSE,
  ...
)
```

Arguments

`db`	the full path to a Pamguard database file or folder of databases
`binaries`	a folder containing Pamguard binary files, all subfolders will also be added
`settings`	an XML settings file from Pamguard
`functions`	a named list of additional functions to add
`verbose`	logical flag to show messages
`default`	logical flag to use default measurement function parameters
`...`	values to pass on to default standardClickCalcs function

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"

`settings` a named list of settings, usually imported from Pamguard's "Export XML Configuration"

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

plotDataExplorer *Explore Data in an Interactive Shiny Plot*

Description

Runs an interactive Shiny 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 event number, detector name, or species

Usage

```
plotDataExplorer(x)
```

Arguments

x	data to plot, can be an AcousticStudy, AcousticEvent, data.frame or a list of AcousticEvent objects

Value

nothing, just plots

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)

if(interactive()) plotDataExplorer(exStudy)
```

plotGram *Plot Spectrogram or Cepstrogram*

Description

Plots either a spectrogram or cepstrogram and also overlays whistle or cepstral contours from the binary files

Usage

```
plotGram(
  x,
  evNum = 1,
  start = NULL,
  end = NULL,
  channel = 1,
  wl = 512,
  hop = 0.25,
  mode = c("spec", "ceps"),
  detections = c("cepstrum", "click", "whistle"),
  detCol = c("red", "blue", "purple"),
  brightness = 0,
  contrast = 0,
  q = 0.01,
  cmap = gray.colors(64, start = 1, end = 0),
  size = 1,
  add = FALSE,
  title = NULL,
  sr = NULL,
  freqRange = NULL,
  ...
)
```

Arguments

x	an AcousticStudy object
evNum	if x is a study, the event index number to plot. Note that this is the index in the order that they appear in the AcousticStudy object, not the actual event number. Alternatively full event names can be used
start	start time of the plot, either POSIXct or seconds from the start of the event
end	end time of the plot, either POSIXct or seconds from the start of the event.
channel	channel to plot
wl	window length of FFT to use for creating plot
hop	hop value of FFT to use for creating plot, either as a percentage of wl or number of samples
mode	one of 'spec' or 'ceps' to plot either spectrogram or cepstrogram
detections	vector containing any of 'cepstrum', 'click', and/or 'whistle' indicating which detections to overlay on the plot
detCol	vector containing colors to use for plotting detections. Order matches order of detections (default alphabetical - cepstrum, click, whistle)
brightness	value from -255 to 255, positive values increase brightness, negative values decrease brightness of concatenated spectrogram image
contrast	value from -255 to 255, positive values increase contrast, negative values decrease contrast of concatenated spectrogram image

q	lower and upper quantiles to remove for scaling concatenated spectrogram. Or if a single value, then quantiles q and 1-q will be used. Ex. if q=.01, then the bottom 1 plotting the image. This is done purely for cosmetic reasons, no output data is affected
cmap	color map for the spectrogram, either a palette function or vector of colors
size	size scaling to apply to plotted points and lines, as a multiple factor to the default values. Can be a vector of length equal to detections to apply different size scales
add	logical flag FALSE to create a new plot (default), or TRUE to only add new detections to an existing plotGram plot with the same parameters. Can be used to add detections with different visual parameters for the same detection type (e.g. clicks with different point size and color). Typically will only work well if start and end have been set manually.
title	optional title for plot, defaults to "Spectrogram" or "Cepstrogram"
sr	sample rate to decimate to
freqRange	frequency range to plot, in Hz for mode='spec' or milliseconds for mode='ceps'
...	additional arguments to pass to points or lines

Value

nothing, just plots

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
recs <- system.file('extdata', 'Recordings', package='PAMpal')
exStudy <- updateFiles(exStudy,
                       bin=system.file('extdata', 'Binaries', package='PAMpal'),
                       db = system.file('extdata', 'Example.sqlite3', package='PAMpal'))
exStudy <- addRecordings(exStudy, folder=recs, log=FALSE, progress=FALSE)
# No detections will appear on plot because included recordings are heavily decimated
plotGram(exStudy)
```

---

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, length = NULL, sr = NULL)

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

plotWigner(x, UID, length = NULL, 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
length	length of the waveform to use for plotting, in samples. The clip used will be centered around the maximum value of the waveform, if length is NULL (default), the entire waveform will be used. If length is greater than the stored clip, the waveform will be zero-padded to length
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'`), or by start and end of recording files (`mode = 'recording'`). Will apply all processing functions in `pps` to the appropriate modules

Usage

```
processPgDetections(
  pps,
  mode = c("db", "time", "recording", "fixed"),
  id = NULL,
  grouping = NULL,
  format = c("%m/%d/%Y %H:%M:%OS", "%m-%d-%Y %H:%M:%OS",
    "%Y/%m/%d %H:%M:%OS", "%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. `time` will organize into events based on timestamps provided in `grouping`. `recording` will organize events by the start and end times of recording files found in the database. For `time` and `recording`, ALL detections between the start and end times are included, for `db` only selected detections are included. `fixed` is similar to `time`, but instead of user-defined start and end times it creates events of a fixed time length (specified by `grouping`) covering the duration of the data.
`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.

For mode = 'fixed' this should be the time duration for the fixed length events. If this is a numeric value, it is the duration in seconds. If it is a character it should be of the format "NumberUnit", e.g. "5min" or "1hour".

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

Details

If mode is not specified, it will try to be automatically determined in the following order. 1) if a grouping data.frame or CSV is provided, then mode='time' will be used. 2) If there are labelled events present in the database, mode='db' will be used. 3) mode='recording' will be used, which should be equivalent to loading all possible data.

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')
# process events by recording event

exStudyRecording <- processPgDetections(exPps, mode='recording', id='Recording')
exFixed <- processPgDetections(exPps, mode='fixed', grouping='1min', id='1Minute')
```

`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)
```

---

removeNote *removeNote*

Description

Remove a note added with addNote

Usage

```
removeNote(x, index)
```

Arguments

x	An AcousticStudy or AcousticEvent object
index	The index of the note to remove, order matches the output of getNotes

Value

For addNote, the same data as x, with notes added. For getNotes, a list of all notes present in x

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
exStudy <- addNote(exStudy, to='study', label='Note1',
                   note='My first note for this study')
exStudy <- addNote(exStudy, to='event', evNum=1:2, label='Note2',
                   note='A note for the first two events')
exStudy
removeNote(exStudy, 1)
removeNote(exStudy, 2)
removeNote(exStudy, 3)
```

---

removeSettings *Remove Settings from a PAMpalSettings Object*

Description

Remove all settings from the "settings" slot in a PAMpalSettings object.

Usage

```
removeSettings(pps)
```

Arguments

pps	a PAMpalSettings object to remove settings from

Value

the same PAMpalSettings object as pps, with all settings removed from the "settings" slot

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
exPps <- new('PAMpalSettings')
exPps <- addSettings(exPps, system.file('extdata', 'Example.xml', package='PAMpal'))
removeSettings(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)
```

runDepthReview *Run Echo Depth Review App*

Description

Runs a Shiny app to review the slant delay and esimated depths of an AcousticStudy object that has been processed with calculateEchoDepth. App allows users to select detections that should not be included in future analysis and marks them with the tag keepClick=FALSE, similar to filterEchoDepths.

Usage

```
runDepthReview(x)
```

Arguments

x	an AcousticStudy object that has been processed with calculateEchoDepth

Value

the object as x, with updated keepClick column

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# example not run because \link{calculateEchoDepth} must be run first,
# and it requires a large amount of data not stored in the package
## Not run:
study <- calculateEchoDepth(study, wav='path/to/wavFiles')
study <- runDepthReview(x)

## End(Not run)
```

---

runIciReview *Run ICI Review App*

Description

Runs a Shiny app that shows three plots - the ICI (time to next detection) over time, histogram of ICI values, and the average waveform of an event. Average waveform plot is only present if calculateEchoDepth has been run first, otherwise IPI and average waveform data will not be present. ICI plots have a red line showing the modal ICI of the event, and average waveform plot has a red line of the estimated IPI level. All plots can be interacted with by clicking on a location to select a new ICI / IPI (called the "User ICI/IPI") that will be shown in blue. "Save ICI/IPI" buttons can be pressed to use save this "User" value as the new All_ici or ipiMax value. "Remove ICI/IPI" buttons can be pressed to set these values to NA, which should be done in cases where no apparent ICI/IPI exists from the plot. The "Reset" button can be used to restore the original modal ICI and estimated IPI values.

Usage

```
runIciReview(x, maxIci = 2.5)
```

Arguments

x	an AcousticStudy object that has been processed with calculateEchoDepth
maxIci	maximum ICI value (seconds) to display on plots

Value

the object as x, with potentially updated All_ici and ipiMax measures for some events depending on user activity

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# average waveform/IPI estimates not present because
# calculateEchoDepth must be run first for those to exist
if(interactive()) {
data(exStudy)
exStudy <- runIciReview(exStudy)
}
```

---

sampleDetector *Subsample Detectors in AcousticStudy*

Description

samples either a fraction or fixed number from each detector in each event of an AcousticStudy

Usage

```
sampleDetector(x, n = 1)
```

Arguments

x	AcousticStudy object
n	if less than 1, proportion of rows to sample from each detector. If 1 or more, the number of rows to sample from each detector.

Details

Uses slice_sample to do the sampling, n is converted either to prop or n based on its size. Negative values are treated the same as in slice_sample

Value

subsampled AcousticStudy x

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
data(exStudy)
nDetections(exStudy)
halfData <- sampleDetector(exStudy, n=0.5)
# there are odd numbers of rows in some detectors, so less than half
nDetections(exStudy)
oneDetPerDetector <- sampleDetector(exStudy, n=1)
nDetections(exStudy)
```

---

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)
```

`summariseDiveDepth` *Summarise Dive Depth*

Description

Summarise results of dive depth estimation using calculateEchoDepth and related functions

Usage

```
summariseDiveDepth(x, hpDepthError = 1, locType = "PGTargetMotion")
```

Arguments

`x`	an AcousticStudy that has been processed with calculateEchoDepth
`hpDepthError`	hydrophone depth error to use for error estimation
`locType`	name of localization, note that this function is not computing any localization, only using previously calculated

Value

a dataframe with columns summarising the estimated dive depth for each event in `x`

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# example not run because \link{calculateEchoDepth} must be run first,
# and it requires a large amount of data not stored in the package
## Not run:
study <- calculateEchoDepth(study, wav='path/to/wavFiles')
summariseDiveDepth(study)

## End(Not run)
```

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 quefrency values

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 500kHz 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

testGPL *A fake GPL detection*

Description

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

Usage

```
data(testGPL)
```

Format

A list with three items:

freq a vector of the frequency contour values in hertz

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

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,
  check = 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
`check`	logical flag to do extra checking. You do not need to set this parameter, used internally for speed purposes so certain checks are not repeated

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)
```

---

updatePamObject *Update PAMpal S4 Object*

Description

Updates older versions of PAMpal's S4 objects to stop "validOjbect" warning messages

Usage

```
updatePamObject(x)
```

Arguments

x	an AcousticStudy, AcousticEvent, or PAMpalSettings object

Details

As of v0.12.0 this updates any previous version's PAMpalSettings objects to have the new "settings" slot, as well as updating any PAMpalSettings objects within an AcousticStudy

Value

the same object as x with any slot changes made

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
## Not run:
pps <- new('PAMpalSettings')
# manually breaking this S4 class, don't try this at home
attr(pps, 'settings') <- NULL
# This will now give an error
pps
pps <- updatePamObject(pps)
# Fixed!
pps

## End(Not run)
```

`writeEventClips` *Create Wav Clips of Data*

Description

Creates audio clips containing sounds from events or detections

Usage

```
writeEventClips(
  x,
  buffer = c(0, 0.1),
  outDir = ".",
  mode = c("event", "detection"),
  channel = 1,
  filter = 0,
  useSample = FALSE,
  progress = TRUE,
  verbose = TRUE,
  fixLength = FALSE
)

parseEventClipName(file, part = c("event", "time", "UID", "channel", "UTC"))
```

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
filter	filter to apply to wav clips before writing, values in kHz. A value of 0 applies no filter. A single value applies a highpass filter at that value. A vector of two values applies a lowpass filter if the first number is 0, or a bandpass filter if both are non-zero.
useSample	logical flag to use startSample information in binaries instead of UTC time for start of detections. This can be slightly more accurate (~1ms) but will take longer
progress	logical flag to show progress bar
verbose	logical flag to show summary messages
fixLength	logical flag to fix the output clip length to a constant value. If TRUE, then output clip length is entirely determined by the buffer value, as if the detection or event had zero length. E.g. buffer=c(-2,1) will produce clips 3 seconds long, starting 2 seconds before the detection/event start time.
file	file name to parse
part	part of file name to return

Details

parseEventClipName parses the file names created to pull out event names or file start times

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]_[Id]CH[ChannelNumber(s)]_[YYYYMMDD]_[HHMMSS]_[mmm].wav The last numbers are the start time of the file in UTC, accurate to milliseconds. The Id is either the event ID or the detection UID.

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