Package ‘PAMscapes’

February 5, 2026

Title Tools for Summarising and Analysing Soundscape Data

Version 0.15.0

Description A variety of tools relevant to the analysis
of marine soundscape data. There are tools for downloading AIS (automatic identification system)
data from Marine Cadastre <https://hub.marinecadastre.gov>,
connecting AIS data to GPS coordinates, plotting summaries of various soundscape
measurements, and downloading relevant environmental variables (wind, swell height) from the
National Center for Atmospheric Research data server <https://gdex.ucar.edu/datasets/d084001/>.
Most tools were developed to work well with output from 'Triton' software, but can be adapted
to work with any similar measurements.

License GNU General Public License

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports dplyr, rlang (>= 1.1.7), ggplot2 (>= 4.0.1), lubridate,
scales, tidyr, httr, data.table, geosphere, sf, PAMmisc (>=
1.12.4), ncdf4, purrr, shiny, future.apply, signal, tuneR, DT

Suggests testthat

NeedsCompilation no

Author Taiki Sakai [aut, cre],
Anne Simonis [ctb],
Shannon Rankin [ctb],
Megan McKenna [ctb],
Kaitlin Palmer [ctb]

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

Repository CRAN

Date/Publication 2026-02-05 01:00:02 UTC

Contents

`addAIS` *Add AIS Data to Dataframe*

Description

Adds matching AIS data downloaded from Marine Cadastre to a dataframe containing location information

Usage

```
addAIS(
  x,
  ais,
  interpType = c("all", "close", "none"),
  interpTime = 0,
  interpCols = NULL
)
```

Arguments

`x`	a dataframe with `UTC`, `Latitude`, and `Longitude` columns
`ais`	AIS data created using the readLocalAIS function
`interpType`	one of `c('all', 'close', 'none')`, the type of time interpolation to apply to `x`. Often the time scale of points in `x` is much longer than the points in `ais`, which can result in awkward looking AIS paths. `'all'` will interpolate all points in `x` to a smaller timescale. `'close'` will interpolate only time ranges in `ais` marked as `inDist` by readLocalAIS. `'none'` will apply no interpolation
`interpTime`	time (seconds) between new `UTC` points. If `0` (default), no interpolation will be done
`interpCols`	names of any extra columns to interpolate (other than `Latitude` and `Longitude`)

Value

a dataframe with AIS data added, will contain more rows than `x` if `ais` has more than one vessel. If any interpolation is applied, any non-constant columns not specified to `interpCols` will be removed

Author(s)

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

Examples

```
gps <- data.frame(Latitude=c(33.2, 33.5,33.6),
                  Longitude=c(-118.1, -118.4, -119),
                  UTC=as.POSIXct(
                    c('2022-04-28 05:00:00',
                      '2022-04-28 10:00:00',
                      '2022-04-28 20:00:00'),
                    tz='UTC'))
ais <- readLocalAIS(gps, aisDir=system.file('extdata/ais', package='PAMscapes'), distance=20e3)
gpsNoInterp <- addAIS(gps, ais, interpType='none')
str(gpsNoInterp)
gpsClose <- addAIS(gps, ais, interpType='close')
str(gpsClose)
gpsAllInterp <- addAIS(gps, ais, interpType='all')
str(gpsAllInterp)
```

`addAISSummary` *Add AIS Data Summary to Dataframe*

Description

Adds a summary of matching AIS data for nearby vessels to a data. Information added includes number of vessels, distance to nearby vessels, and average speed of nearby vessels

Usage

```
addAISSummary(x, ais, distance = 10000)
```

Arguments

`x`	a dataframe with `UTC`, `Latitude`, and `Longitude` columns
`ais`	AIS data created using the readLocalAIS function. Can also be a character listing the directory of AIS
`distance`	distance (meters) within locations in `x` to mark as "nearby"

Value

a dataframe with AIS summary data added. Will contain new columns

nShips the number of ships within "distance" at this time

meanDist average distance of nearby ships, NA if none

meanSOG average speed over ground of nearby ships, NA if none

closeDist distance of the closest ship, NA if none

closeSOG speed over ground of closest ship, NA if none

Author(s)

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

Examples

```
gps <- data.frame(Latitude=c(33.2, 33.5,33.6),
                  Longitude=c(-118.1, -118.4, -119),
                  UTC=as.POSIXct(
                    c('2022-04-28 05:00:00',
                      '2022-04-28 10:00:00',
                      '2022-04-28 20:00:00'),
                    tz='UTC'))
ais <- readLocalAIS(gps, system.file('extdata/ais', package='PAMscapes'))
aisSummary <- addAISSummary(gps, ais)
str(aisSummary)
```

binDetectionData *Bin Detection Data to Time Bins*

Description

Transforms detection data to presence-type data with user specified time bin (e.g. hourly or daily presence).

Usage

```
binDetectionData(
  x,
  bin,
  columns = c("species"),
  rematchGPS = TRUE,
  gpsGroup = NULL
)
```

Arguments

`x`	dataframe of detection data
`bin`	the amount time to bin by, must be a character of the form `"#unit"` or `"unit"` e.g. "2hour" or "day"
`columns`	names of the columns in `x` that define which rows should still be considered distinct even if their times are in the same bin. For example, two calls from the same species in one hour should result in one row of hourly presence, but two calls from different species in one hour should result in two separate rows of hourly presence.
`rematchGPS`	logical flag, if `TRUE` then if columns Longitude and Latitude are present in `x` then they will be rematched to the outputs. Note that this is imprecise - the time used for rematching the outputs is the center of each output time bin.
`gpsGroup`	the name of the column in `x` that denotes different GPS groupings within the data, usually something like "site" or "deployment." Not needed if all data are from the same location.

Value

a dataframe where each row represents detection presence of one time unit

Author(s)

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

Examples

```
dets <- data.frame(
  UTC = as.POSIXct(c('2020-04-04 12:20:00', '2020-04-04 12:40:00', '2020-04-04 13:20:00')),
   species = c('whale', 'whale', 'dolphin'),
   call = c('a', 'b', 'c'))
# two rows of outputs
binDetectionData(dets, bin='1hour', columns='species')
# adding "call" creates 3 rows of outputs
binDetectionData(dets, bin='1hour', columns=c('species', 'call'))
```

---

binSoundscapeData *Summarise Soundscape Data by Time Bin*

Description

Bins soundscape measurements by a unit of time and summarises them using a function (usually the median)

Usage

```
binSoundscapeData(
  x,
  bin = "1hour",
  method = c("median", "mean"),
  binCount = FALSE,
  extraCols = NULL
)
```

Arguments

`x`	a data.frame of soundscape metric data read in with loadSoundscapeData
`bin`	amount of time to bin data by, format can be "#Unit" e.g. `'2hour'` or `'1day'`
`method`	summary function to apply to data in each time bin, must be one of "median" or "mean"
`binCount`	logical flag to return the number of times in each time bin as column "binCount"
`extraCols`	Additional non-frequency columns in `x` to apply the binning to

Value

a summarised version of the input data.frame `x`

Author(s)

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

`checkSoundscapeInput` *Check Proper Formatting for Soundscape Inputs*

Description

Reads and checks data to ensure formatting will work for other `PAMscapes` functions. Will read and check the formatting of CSV files, or check the formatting of dataframes. Can also read in MANTA NetCDF files and format the data appropriately.

Usage

```
checkSoundscapeInput(
  x,
  needCols = c("UTC"),
  skipCheck = FALSE,
  timeBin = NULL,
  binFunction = median,
  octave = c("original", "tol", "ol"),
  label = NULL,
  tz = "UTC",
  extension = c("nc", "csv")
)
```

Arguments

`x`	a dataframe, path to a CSV file, or path to a MANTA NetCDF file, or folder containing these. If `x` is a vector of file paths then all will be read in and combined. If `x` is a folder, then all files with extension `extension` will be loaded. Note this will not load files within subfolders, only the main folder.
`needCols`	names of columns that must be present in `x`, if any are missing will trigger an error
`skipCheck`	logical flag to skip some data checking, recommended to keep as `FALSE`
`timeBin`	amount of time to bin data by, format can be "#Unit" e.g. `'2hour'` or `'1day'`
`binFunction`	summary function to apply to data in each time bin
`octave`	one of "original", "tol", or "ol". If "original" then nothing happens, otherwise data are converted to Octave-leve ("ol") or Third-Octave-Level ("tol") measurements using createOctaveLevel
`label`	optional, if not `NULL` then this value will be added as an additional column "label" to the output
`tz`	timezone of the data being loaded, will be converted to UTC after load
`extension`	only used if `x` is a folder, the file extension to load. Must be one of "nc" or "csv"

Details

Files created by MANTA and Triton software will be reformatted to have consisitent formatting. The first column will be renamed to "UTC", and columns containing soundscape metrics will be named using the convention "TYPE_FREQUENCY", e.g. "HMD_1", "HMD_2" for Manta hybrid millidecade mesaurements.

Inputs from sources other than MANTA or Triton can be accepted in either "wide" or "long" format. Wide format must follow the conventions above - first column "UTC", other columns named by "TYPE_FREQUENCY" where TYPE is consistent across all columns and FREQUENCY is in Hertz. Long format data must have the following columns:

"UTC" - time of the measurement, in UTC timezone

"type" - the type of soundscape measurement e.g. PSD or OL, must be the same for all

"frequency" - the frequency of the measurement, in Hertz

"value" - the soundscape measurement value, usually dB

Value

a dataframe

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
manta <- checkSoundscapeInput(system.file('extdata/MANTAExampleSmall1.csv', package='PAMscapes'))
str(manta)
ol <- checkSoundscapeInput(system.file('extdata/OLSmall.csv', package='PAMscapes'))
str(ol)
psd <- checkSoundscapeInput(system.file('extdata/PSDSmall.csv', package='PAMscapes'))
str(psd)
```

`createOctaveLevel` *Create Octave Level Measurements*

Description

Creates (third) octave level or broadband measurements from finer resolution soundscape metrics, like Power Spectral Density (PSD) or Hybrid Millidecade (HMD) measures

Usage

```
createOctaveLevel(
  x,
  type = c("ol", "tol", "broadband", "bb", "hmd"),
  freqRange = NULL,
  normalized = FALSE
)
```

Arguments

`x`	dataframe of soundscape metrics
`type`	one of `'ol'` to create octave level, `'tol'` to create third octave level measures, or `'broadband'` or `'bb'` to create an arbitrary broadband measure. For broadband measures, `freqRange` must be supplied to define the range
`freqRange`	a vector of the minimum and maximum center frequencies (Hz) desired for the output. If `NULL`, full available range of frequencies will be used. If output `type` is broadband, this is used to define the lower and upper bounds of the desired output broadband level
`normalized`	logical flag to return values normalized by the bandwidth of each octave level band (per Hz)

Details

To create new measurements, finer resolution metrics are cast to linear space, summed, and then re-logged. If input measurements are HMD values then they are assumed to be normalized per Hz, so levels are first corrected by the bandwidth before summing. In all other cases inputs are assumed to not be normalized per Hz measurements and are just summed.

Value

a dataframe with summarised octave level band measurements

Author(s)

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

Examples

```
psd <- loadSoundscapeData(system.file('extdata/PSDSmall.csv', package='PAMscapes'))
str(psd)
tol <- createOctaveLevel(psd, type='tol')
str(tol)
ol <- createOctaveLevel(tol, type='ol')
str(ol)
bb <- createOctaveLevel(psd, type='bb', freqRange=c(20, 150))
str(bb)
```

`downloadMarCadAIS` *Download AIS Data from Marine Cadastre*

Description

Downloads daily AIS files from https://hub.marinecadastre.gov/pages/vesseltraffic covering the date range present in input data

Usage

```
downloadMarCadAIS(x, outDir, overwrite = FALSE, unzip = TRUE, verbose = TRUE)
```

Arguments

`x`	a dataframe with column `UTC` in POSIXct format
`outDir`	directory to save the downloaded files
`overwrite`	logical flag to overwrite existing data. Recommended to be `FALSE` to avoid re-downloading large files unnecessarily
`unzip`	logical flag to unzip downloaded files. Original downloads from Marine Cadastre come as large .zip
`verbose`	logical flag to print messages about download progress

Value

a vector of the paths to the downloaded .zip files, any days that were unable to download will be `NA`

Author(s)

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

Examples

```
gps <- data.frame(Latitude=c(33.2, 33.5,33.6),
                  Longitude=c(-118.1, -118.4, -119),
                  UTC=as.POSIXct(
                    c('2022-04-28 05:00:00',
                      '2022-04-28 10:00:00',
                      '2022-04-28 20:00:00'),
                    tz='UTC'))
tempDir <- tempdir()
# Commented out because running this will download
# a ~500mb file
# marcadFiles <- downloadMarCadAIS(gps, outDir=tempDir)
```

`evaluateDeployment` *Evaluate deployment of recording files for potential problems*

Description

Runs a number of quality assurance / quality control (QAQC) checks on a folder of recording files to identify potential problems. These include checking the start and end times of files for consistency to identify potential data gaps, measuring sound levels in each file to identify potential recorder issues, and (if applicable) checking battery and temperature data to identify potential instrument failure. Can also create spectrogram images throughout the deployment to aid in visually checking for problems or noise.

Usage

```
evaluateDeployment(
  dir,
  excludeDirs = c("Post_Retrieval_Data", "Pre_Deployment_Data"),
  sampleWindow = c(60, 120),
  channel = 1,
  sensitivity = NA,
  calibration = NULL,
  timeRange = NULL,
  name = NULL,
  subDirPattern = NULL,
  outDir = NULL,
  nSpectrograms = 0,
  specLength = 1800,
  panelLength = 300,
  log = FALSE,
  progress = TRUE,
  verbose = TRUE
)
```

Arguments

`dir`	folder or folders containing recordings and optionally Soundtrap .log.xml files. All .wav and .log.xml files within `dir` will be analysed, as well as all files in each subfolder of `dir` (only going down one level).
`excludeDirs`	the names of any subfolders within `dir` that should be excluded (e.g. if `dir` contains folders "Recordings" and "Clips" then `excludeDirs="Clips"` would result in only the "Recordings" folder being analysed
`sampleWindow`	start and end (in seconds) of the time window to use for analysis, e.g. `c(40, 100)` will use a 60 second window starting 40 seconds into the file
`channel`	channel number of recording files to use for analysis
`sensitivity`	the sensitivity of the recording device in dB, this is typically a large negative number
`calibration`	if not `NULL`, the frequency dependent calibration to apply. Must have "frequency" and "gain" (in dB), can either be a .tf file, a CSV file with columns for frequency and gain, a dataframe with columns frequency and gain, or a NetCDF with "frequency" dimension and "senstivity" or "gain" variable
`timeRange`	if not `NULL`, a vector of two POSIXct times identifying the expected start and end times of the deployment. If the actual start and end times of of the recording files are earlier or later than these, then a warning will be issued and no calculations will be done, returning `NULL`
`name`	a name to assign for this deployment, used for plot labeling, logging, and stored as `projectName` with the output dataframe. If left as `NULL`, then the basename of `dir` will be used.
`subDirPattern`	if not `NULL`, a pattern to use for selecting which subfolders of `dir` to use for analysis. E.g. if folders "Site1_Recordings" and "Site2_Recordings" both exist

	in `dir`, then `subDirPattern="^Site1"` would result in analysing only the first folder of recordings
`outDir`	if not `NULL`, a directory to store outputs. Outputs include plots, a CSV of the calculated metrics, and a .txt log file if `log=TRUE`
`nSpectrograms`	number of spectrogram images to generate. Recording files for creating the spectrograms will be approximately evenly spaced throughout the deployment - if `nSpectrograms=5`, then the first recording will be randomly chosen from the first 20% of files, the second from the second 20%, etc.
`specLength`	length of spectrograms to create, in seconds
`panelLength`	length of each panel of the spectrogram plot, in seconds. This must be less than `specLength`, and is used to avoid creating awkwardly long plots. If `specLength=360` and `panelLength=60`, then the result will be a 6 panel plot where each section is 60 seconds long
`log`	if `TRUE` and `outDir` is not `NULL`, then a text file named "(name)_EvaluateRecorder_LogFile.txt" will be created in `outDir` logging progress and warning messages
`progress`	logical flag to show a progress bar
`verbose`	logical flag to show some messages

Value

a dataframe of the QAQC metric outputs for each recording file

---

`evaluateRecordings` *Evaluate Recording Files for Issues*

Description

Evaluates recording files for potential problems. Sound levels are calculated for a small section of each recording file, this is typically done to check for recorder malfunction. Additionally times between the starts and ends of files are calculated, this is typically done to check for gaps in data.

Usage

```
evaluateRecordings(
  wavFiles,
  sampleWindow = c(60, 120),
  octave = c("tol", "ol"),
  channel = 1,
  freqRange = NULL,
  calibration = NULL,
  sensitivity = 0,
  progress = TRUE
)
```

Arguments

`wavFiles`	file paths to wav files to evaluate, or the directory containing the wav files
`sampleWindow`	start and end (in seconds) of the time window to use for analysis, e.g. `c(40, 100)` will use a 60 second window starting 40 seconds into the file
`octave`	type of sound level to calculate, either `'tol'` for third octave level or `'ol'` for octave level
`channel`	channel of the file to use for analysis
`freqRange`	if not `NULL`, a vector of two numbers giving the range of frequencies to use for analysis (`NULL` will use the full available range)
`calibration`	if not `NULL`, the frequency dependent calibration to apply. Must have "frequency" and "gain" (in dB), can either be a .tf file, a CSV file with columns for frequency and gain, a dataframe with columns frequency and gain, or a NetCDF with "frequency" dimension and "senstivity" or "gain" variable
`sensitivity`	the sensitivity of the recording device in dB, this is typically a large negative number
`progress`	logical flag to show a progress bar

Value

a dataframe containing the sound level and data gap measurements for each file

Author(s)

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

`formatEffort` *Format Detection Effort*

Description

Format effort data for use in other acoustic detection plotting functions. Time ranges will be marked as either "on" or "off" effort

Usage

```
formatEffort(
  effort,
  range = NULL,
  resolution = NULL,
  columns = NULL,
  combineYears = FALSE
)
```

Arguments

`effort`	dataframe with columns `start` or `effortStart` and `end` or `effortEnd` describing on effort time ranges
`range`	if not `NULL`, the full extent time ranges to consider for marking off effort times
`resolution`	if not `NULL`, time resolution to round effort start and end times to. Start times will use floor_date and end times will use ceiling_date, must be a character that is valid for the `unit` argument of those functions
`columns`	if not `NULL`, extra columns to use for differentiating different types of effort that should be tracked separately (e.g. different deployment sites or species with different effort)
`combineYears`	logical flag to combine all years into a single "year"

Value

a dataframe with columns `start`, `end`, and `status` which is either "on" or "off", as well as any columns listed in `columns`

Author(s)

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

`loadDetectionData` *Load Detection Data*

Description

Loads and formats detection data into a common format for use in other PAMscapes functions

Usage

```
loadDetectionData(
  x,
  source = c("makara", "csv"),
  columnMap = NULL,
  detectionType = c("auto", "presence", "detection"),
  presenceDuration = NULL,
  dateFormat = c("%Y-%m-%dT%H:%M:%S%z", "%Y-%m-%d %H:%M:%S",
    "%m-%d-%Y %H:%M:%S", "%Y/%m/%d %H:%M:%S", "%m/%d/%Y %H:%M:%S"),
  tz = "UTC",
  wide = FALSE,
  speciesCols = NULL,
  detectedValues = NULL,
  extraCols = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	dataframe or path to CSV file containing detection data
`source`	source of the detection data, choices other than "csv" just specify specific formatting options
`columnMap`	a list or data.frame specifying how to map the input column names to the required standard names of "UTC", "end", and "species". If a list, must be a named list where the names are the standardized column names and the values are the existing names, e.g. `list('UTC'='start', 'species'='SpeciesName')`. If a data.frame, must have columns "old" with the existing column names and "new" with the standardized name to change it to. All columns successfully changed will be kept with the output
`detectionType`	one of "auto", "presence", or "detection" specifying the type of detection in the data. "presence" means hourly or daily presence style of detections - the duration of the detection is used for the time unit (e.g. hourly presence might have "UTC" value 2020-01-01 12:00:00 and "end" value 2020-01-01 13:00:00 for a detection). "detection" means the data refer to specific detections or bouts of detections rather than just presence. "auto" means that the type of detection will be inferred from the start and end time of each detection - any detections with a duration of exactly one hour or exactly one day will be marked as "presence", any other duration will be marked as "detection"
`presenceDuration`	if `detectionType='presence'`, the duration in seconds, e.g. 86400 for daily presence. Alternative can be a character of the form "(NUMBER)(DURATION)" e.g "2hour" or "1day"
`dateFormat`	format string of dates, see strptime. Can be a vector of multiple formats
`tz`	time zone of input data
`wide`	logical flag indicating whether the input data has species detection information in wide (instead of long) format. If `TRUE`, then this means that there are multiple columns representing multiple kinds of detections, e.g. one column for each different species present. If `FALSE`, then there is a single column that indicates what kind of detection it is.
`speciesCols`	only used if `wide=TRUE`, the names of the columns containing the different types of detections
`detectedValues`	only used if `wide=TRUE`, the values in each `speciesCols` column that indicate a positive detection. e.g. if "0" represents no detection and "1" represents a detection, then this should be "1". Note that all values will be converted to characters, so the string `"1"` must be used instead of the numeric `1`
`extraCols`	(optional) any additional columns to keep with the output
`verbose`	logical flag to show messages
`...`	additional arguments used for certain `source` values

Value

a dataframe with columns UTC, end, species, and detectionType, where each row represents a single detection event. May have additional columns depending on other parameters

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
detFile <- system.file('extdata/detectionExample.csv', package='PAMscapes')
detData <- loadDetectionData(
              detFile, source='csv',
              columnMap=list(UTC='start',end='end', species='sound_code'))
```

---

loadMantaNc *Load MANTA NetCDF File*

Description

Reads in hybrid millidecade data from a MANTA NetCDF output file and formats it into the dataframe format required for use in other PAMscapes functions

Usage

```
loadMantaNc(x, keepQuals = c(1), keepEffort = TRUE)
```

Arguments

`x`	path to .nc file
`keepQuals`	quality flag values to keep. Accepts vector of integers from (1, 2, 3, 4) corresponding to flag labels "Good", "Not evaluated/Unknown", "Compromised/Questionable", and "Unusable/Bad". HMD levels for points with data quality flags outside of `keepQuals` will be marked as `NA`.
`keepEffort`	if `TRUE` or `FALSE`, a logical flag whether or not to keep the effort information with the outputs (number of seconds per minute). If a numeric value, then any minutes with an effort value less than `keepEffort` will be removed (e.g. `50` will remove minutes with less than 50 seconds of effort)

Value

a dataframe with first column UTC and other columns named HMD_Frequency

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# no sample NetCDF provided (too large)

manta <- loadMantaNc('MANTA.nc')
```

`loadMultiscapeData` *Load Multiple Folders of Soundscape Data*

Description

Loads soundscape data just like loadSoundscapeData, but is designed to load multiple soundscape datasets from multiple folders. This is identical to loading each folder of data individually with the same bin and label parameters.

Usage

```
loadMultiscapeData(
  x,
  timeBin = NULL,
  binFunction = "median",
  binCount = FALSE,
  octave = c("original", "tol", "ol"),
  label = NULL,
  keepQuals = c(1),
  keepEffort = TRUE,
  dropNonHmd = TRUE,
  tz = "UTC",
  extension = c("nc", "csv")
)
```

Arguments

`x`	a vector of folder names to load
`timeBin`	amount of time to bin data by, format can be "#Unit" e.g. `'2hour'` or `'1day'`. Unlike loadSoundscapeData this argument is now mandatory to reduce data size
`binFunction`	summary function to apply to data in each time bin, default is "median"
`binCount`	logical flag to return the number of times in each time bin as column "binCount"
`octave`	one of "original", "tol", or "ol". If "original" then nothing happens, otherwise data are converted to Octave-leve ("ol") or Third-Octave-Level ("tol") measurements using createOctaveLevel
`label`	if not `NUL`, then must be of equal length to `x`

`keepQuals`	quality flag values to keep. Accepts vector of integers from (1, 2, 3, 4) corresponding to flag labels "Good", "Not evaluated/Unknown", "Compromised/Questionable", and "Unusable/Bad". HMD levels for points with data quality flags outside of `keepQuals` will be marked as `NA`.
`keepEffort`	if `TRUE` or `FALSE`, a logical flag whether or not to keep the effort information with the outputs (number of seconds per minute). If a numeric value, then any minutes with an effort value less than `keepEffort` will be removed (e.g. `50` will remove minutes with less than 50 seconds of effort)
`dropNonHmd`	logical flag to drop non-standard hybrid millidecade bands, only applies to HMD type data. Some datasets have frequency values that are not part of the standard HMD bands (e.g. at exactly the Nyquist rate), if `TRUE` these will be removed.
`tz`	timezone of the data being loaded, will be converted to UTC after load
`extension`	only required if both netCDF and CSV files exist in the folders to load, in which case only one type will be loaded. Must be one of "nc" or "csv"

Details

This function is equivalent to loading each folder of data separately with the same time and octave-level aggregation options applied, and is meant as a convenient wrapper for loading multiple years or sites of data for comparison person. The expectation is that this function will be primarily used for large scale comparisons, hence why `timeBin` is a required argument to reduce data resolution.

The only other difference is that if no labels are supplied for the folders, then one will be generated either from the names of `x` if it is a named vector, or the name of the folder using basename. This is to ensure that each separate folder can be identified once read in.

Value

a dataframe

Author(s)

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

`loadSoundscapeData` *Load Soundscape Data*

Description

Reads and checks data to ensure formatting will work for other `PAMscapes` functions. Will read and check the formatting of CSV files, or check the formatting of dataframes. Can also read in MANTA NetCDF files and format the data appropriately.

Usage

```
loadSoundscapeData(
  x,
  needCols = c("UTC"),
  skipCheck = FALSE,
  timeBin = NULL,
  binFunction = "median",
  binCount = FALSE,
  octave = c("original", "tol", "ol"),
  label = NULL,
  keepQuals = c(1),
  keepEffort = TRUE,
  dropNonHmd = TRUE,
  tz = "UTC",
  extension = c("nc", "csv")
)
```

Arguments

`x`	a dataframe, path to a CSV file, or path to a MANTA NetCDF file, or folder containing these. If `x` is a vector of file paths then all will be read in and combined. If `x` is a folder, then all files with extension `extension` will be loaded. Note this will not load files within subfolders, only the main folder.
`needCols`	names of columns that must be present in `x`, if any are missing will trigger an error
`skipCheck`	logical flag to skip some data checking, recommended to keep as `FALSE`
`timeBin`	amount of time to bin data by, format can be "#Unit" e.g. `'2hour'` or `'1day'`
`binFunction`	summary function to apply to data in each time bin, default is "median"
`binCount`	logical flag to return the number of times in each time bin as column "binCount"
`octave`	one of "original", "tol", or "ol". If "original" then nothing happens, otherwise data are converted to Octave-leve ("ol") or Third-Octave-Level ("tol") measurements using createOctaveLevel
`label`	optional, if not `NULL` then this value will be added as an additional column "label" to the output
`keepQuals`	quality flag values to keep. Accepts vector of integers from (1, 2, 3, 4) corresponding to flag labels "Good", "Not evaluated/Unknown", "Compromised/Questionable", and "Unusable/Bad". HMD levels for points with data quality flags outside of `keepQuals` will be marked as `NA`.
`keepEffort`	if `TRUE` or `FALSE`, a logical flag whether or not to keep the effort information with the outputs (number of seconds per minute). If a numeric value, then any minutes with an effort value less than `keepEffort` will be removed (e.g. `50` will remove minutes with less than 50 seconds of effort)
`dropNonHmd`	logical flag to drop non-standard hybrid millidecade bands, only applies to HMD type data. Some datasets have frequency values that are not part of the standard HMD bands (e.g. at exactly the Nyquist rate), if `TRUE` these will be removed.

`tz`	timezone of the data being loaded, will be converted to UTC after load
`extension`	only used if `x` is a folder, the file extension to load. Must be one of "nc" or "csv"

Details

Files created by MANTA and Triton software will be reformatted to have consisitent formatting. The first column will be renamed to "UTC", and columns containing soundscape metrics will be named using the convention "TYPE_FREQUENCY", e.g. "HMD_1", "HMD_2" for Manta hybrid millidecade mesaurements.

Inputs from sources other than MANTA or Triton can be accepted in either "wide" or "long" format. Wide format must follow the conventions above - first column "UTC", other columns named by "TYPE_FREQUENCY" where TYPE is consistent across all columns and FREQUENCY is in Hertz. Long format data must have the following columns:

"UTC" - time of the measurement, in UTC timezone

"type" - the type of soundscape measurement e.g. PSD or OL, must be the same for all

"frequency" - the frequency of the measurement, in Hertz

"value" - the soundscape measurement value, usually dB

Value

a dataframe

Author(s)

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

Examples

```
manta <- loadSoundscapeData(system.file('extdata/MANTAExampleSmall1.csv', package='PAMscapes'))
str(manta)
ol <- loadSoundscapeData(system.file('extdata/OLSmall.csv', package='PAMscapes'))
str(ol)
psd <- loadSoundscapeData(system.file('extdata/PSDSmall.csv', package='PAMscapes'))
str(psd)
```

`markNA` *Mark NA Values by Time and Frequency*

Description

Marks values within a soundscape dataframe as NA according to provided time and (optionally) frequency values

Usage

```
markNA(x, na, by = NULL)
```

Arguments

`x`	dataframe of soundscape data to mark NAs in
`na`	dataframe listing areas to mark NA. Must have columns `start` and `end` in UTC listing time ranges. Can also have columns `freqMin` and `freqMax` to also have accompanying frequency ranges, otherwise all frequency values within the time range will be set to NA
`by`	optional column name in both `x` and `na` if only certain rows of `na` should apply to certain rows of `x` (e.g. if these contain multiple deployments overlapping in time, a "DeploymentName" column can be used to only mark appropriate times)

Value

same dataframe as `x` but with some values replaced with `NA`

Author(s)

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

Examples

```
manta <- loadSoundscapeData(system.file('extdata/MANTAExampleSmall1.csv', package='PAMscapes'))
naDf <- data.frame(start=min(manta$UTC),
                   end=max(manta$UTC),
                   freqMin=100,
                   freqMax=500)
plotHourlyLevel(manta)
plotHourlyLevel(markNA(manta, na=naDf))
```

`matchDetectionData` *Match Detections to Other Data*

Description

Matches detection data from loadDetectionData to other data types. A new column is created in the input data to store detection data, and values in that column are filled in based on whether or not their times overlap with times in the detection data.

Usage

```
matchDetectionData(
  x,
  detection,
  name,
  value,
  fillNA = NA,
  by = NULL,
  mode = c("replace", "add")
)
```

Arguments

`x`	a dataframe with column "UTC" and optionally "end"
`detection`	dataframe of detection data loaded with loadDetectionData
`name`	name of the column in `x` to add detection information to, will be created if it does not exist
`value`	value to use for marking detections. If `value` is the name of a column in `detection`, then the corresponding value of that column will be used. If not, then `value` will be used for all detections
`fillNA`	value to fill in for times that do not have detections, default is `NA`
`by`	if not `NULL`, a specification for pairing distinct groups in `x` and `detection`. Can be a single character of the column name shared by both (e.g. `"site"`), a named character where the name is the column in `x` and the value is the column in `detection` (e.g. `c("site"="deployment")`), or an equality statement where the left is the column in `x` and the right is the column in `detection` (e.g. `"site" == "deployment"`)
`mode`	one of "replace" or "add", this sets how to deal with times that have multiple detections or where column `name` has an existing value. "replace" always replaces with the most recent value in `detection`, "add" concatenates all values with ","

Value

a dataframe

Author(s)

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

Examples

```
time <- as.POSIXct('2020-01-01 12:00:00', tz='UTC')
data <- data.frame(UTC=time+c(0, 30, 60, 90, 120),
                   end=time+c(30, 60, 90, 120, 150))
detection <- data.frame(UTC=time+c(25, 90),
                        end=time+c(40, 95),
                        species=c('A', 'B'))

# Create a new "presence" column and fill with TRUE
matchDetectionData(data, detection, name='presence', value=TRUE)
# Fill non-matching times with FALSE (instead of default NA)
matchDetectionData(data, detection, name='presence', value=TRUE, fillNA=FALSE)
# Instead fill with value from "species" column
matchDetectionData(data, detection, name='presence', value='species')

detection <- data.frame(UTC=time+c(25, 90, 18, 30),
                        end=time+c(40, 95, 28, 40),
                        site=c('east', 'east', 'west', 'west'),
                        species=c('A', 'B', 'A', 'B'))
```

```
data <- data.frame(UTC=time+c(0, 30, 60, 90, 120, 0, 30),
                   end=time+c(30, 60, 90, 120, 150, 30, 60),
                   deployment=c(rep('east', 5), rep('west', 2)))

# detection now has overlapping times, so this will trigger a warning
matchDetectionData(data, detection, name='presence', value='species')
# mode='add' will change this to concatenate the labels for overlaps
matchDetectionData(data, detection, name='presence', value='species', mode='add')
# but really these correspond to different locations, so we can use "by" for that
matchDetectionData(data, detection, name='presence', value='species', by=c('deployment'='site'))
# this is another way to specify "by"
matchDetectionData(data, detection, name='presence', value='species', by='deployment'=='site')
```

---

matchGFS — *Match GFS Environmental Data*

Description

Downloads and matches wind and precipitation data from the Global Forecast System (GFS) weather model. Data is downloaded from the National Center for Atmospheric Research data server https://gdex.ucar.edu/datasets/d084001/. The particular GFS dataset downloaded is the closest "forecast" dataset to the particular time (e.g. .f000 or .f003)

Usage

```
matchGFS(x, progress = TRUE, keepMatch = TRUE)
```

Arguments

`x`	a dataframe with columns `UTC`, `Latitude` and `Longitude` to add environmental data to
`progress`	logical flag to display download progress
`keepMatch`	logical flag to keep the "matchLat", "matchLong", and "matchTime" columns with the output. These are only used to verify which coordinates within the NetCDF were matched to your data.

Value

a dataframe with wind (m/s) and precipitation rate (kg/m^2/s) columns added:

windU Eastward wind velocity

windV Northward wind velocity

windMag Total wind magnitude

precRate Precipitation rate

matchLat Cosest latitude coordinate matched in GFS

matchLong Closest longitude coordinate matched in GFS

matchTime Closest time coordinate matched in GFS

Where the last three columns are only included if `keepMatch=TRUE`

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
# API response may be slow for this example

gps <- data.frame(Latitude=c(33.2, 33.5,33.6),
                  Longitude=c(-118.1, -118.4, -119),
                  UTC=as.POSIXct(
                    c('2022-04-28 05:00:00',
                      '2022-04-28 10:00:00',
                      '2022-04-28 20:00:00'), tz='UTC'))
gps <- matchGFS(gps)
```

`matchSeascape` *Match Seascape Class to Data*

Description

Downloads and matches relevant Seascape class data from the ERDDAP (Environmental Research Division's Data Access Program) server at `https://cwcgom.aoml.noaa.gov/erddap/index.html`. More information on theclasses can be found on the help page for the seascapeR package `https://marinebon.org/seascapeR/index.html`.

Usage

```
matchSeascape(x, type = c("monthly", "8day"), progress = TRUE)
```

Arguments

`x`	a dataframe with columns `UTC`, `Latitude` and `Longitude` to add environmental data to
`type`	the type of seascape data to download, one of "monthly" or "8day"
`progress`	logical flag whether or not to show download progress

Details

This function is just a wrapper around matchEnvData pointing to the specific base URL and dataset ID relevant for seascape data

Value

the same dataframe as `x`, but with new columns `seascapeClass` and `seascapeProb` representing the "CLASS" and "P" variables from the dataset

Author(s)

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

`plotAcousticScene` *Plot Acoustic Scene*

Description

Plots a representation of the acoustic scene using detections in data. Frequency ranges for detections are taken from user input and displayed as different colored bars

Usage

```
plotAcousticScene(
  x,
  freqMap = NULL,
  typeCol = "species",
  title = NULL,
  bin = "1day",
  by = NULL,
  combineYears = FALSE,
  effort = NULL,
  scale = c("log", "linear"),
  freqMin = NULL,
  freqMax = NULL,
  fill = TRUE,
  alpha = 1,
  returnData = FALSE,
  add = FALSE
)
```

Arguments

`x`	dataframe of detections, must have column `UTC` and a column to connect detection types to the frequency type map
`freqMap`	a dataframe listing frequency ranges to use for various detection types in `x`. Must have columns `type`, `freqMin` (Hz), `freqMax` (Hz), and optionally `color` (color to use for this type of detection on plot)
`typeCol`	column name in `x` that matches names in `type` column in `freqMap`
`title`	optional title to use for the plot
`bin`	time bin to use for plotting time axis. Each detection will be displayed as covering this amount of time
`by`	if not `NULL`, column name to facet plot by (e.g. site)
`combineYears`	logical flag to combine all observations to display as a single "year". The year will be set to 2019, and detections falling on leap days (February 29th) will be removed

`effort`	if not `NULL`, a dataframe decribing effort data to be be formatted with formatEffort. Alternatively, if columns "effortStart" and "effortEnd" are present in `x`, then these will be used.
`scale`	one of `log` or `linear`, the frequency scale for the plot
`freqMin`	optional minimum frequency for plot, useful for log scale
`freqMax`	optional maximum frequency for plot
`fill`	logical flag if `TRUE` then filled boxes will be plotted, if `FALSE` then only outlines will be plotted
`alpha`	transparency percentage for plotting, values less than 1 will allow multiple overlapping colors to be seen
`returnData`	if `TRUE` then no plot will be generated, instead the dataframe that would normally be used to make the plot will be returned
`add`	logical flag if `FALSE` plots normally if `TRUE` then the output can be (maybe) added to an existing ggplot object

Value

a ggplot object

Author(s)

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

Examples

```
detDf <- data.frame(
   UTC=as.POSIXct(c('2023-01-01 00:00:00',
                    '2023-01-03 00:00:00',
                    '2023-01-02 12:00:00',
                    '2023-01-04 00:00:00'),
                  tz='UTC'),
  species = c('Dolphin', 'Whale', 'Whale', 'Dolphin'))
freqMap <- data.frame(type=c('Dolphin', 'Whale'),
                      freqMin=c(10e3, 100),
                      freqMax=c(30e3, 400),
                      color=c('darkgreen', 'blue'))
plotAcousticScene(detDf, freqMap=freqMap, typeCol='species', bin='1day')

detFile <- system.file('extdata/detectionExample.csv', package='PAMscapes')
detData <- loadDetectionData(detFile, source='csv',
                     columnMap=list(UTC='start', end='end', species='sound_code'))
plotAcousticScene(detData)
```

`plotDetectionBoxplot` *Plot Detection Boxplot*

Description

Plots time series of boxplots showing detection data across time

Usage

```
plotDetectionBoxplot(
  x,
  group = "species",
  facet = NULL,
  color = hue_pal(),
  bin = "day/week",
  combineYears = FALSE,
  effort = NULL,
  dropZeroes = FALSE,
  title = NULL,
  returnData = FALSE
)
```

Arguments

`x`	dataframe of detection data read in with loadDetectionData
`group`	name(s) of columns indicating which rows of `x` are distinct from each other, typically something like "site" or "species" or both. These are used to define the different data points that go in to each boxplot, see Details below for more information.
`facet`	if not `NULL`, name of the column in `x` to facet the plot by
`color`	only used if `facet` is not `NULL`, colors to use for each separate facet. Can either be a color palette function or a character vector of color names. If a vector, it can be named by the levels in `facet` that each color should correspond to
`bin`	time bins to use for generating plot, must be a character of format "time1/time2" where "time1" will be the y-axis of the plot and "time2" will be the x-axis of the plot. Times are one of "hour", "day", "week", or "month" (e.g. `"day/week"`).
`combineYears`	logical flag to combine all observations to display as a single "year"
`effort`	if not `NULL`, a dataframe describing on effort times to be formatted with formatEffort. If effort data is not provided then times with zero detections will not be properly accounted for. Alternatively, if columns "effortStart" and "effortEnd" are present in `x`, then these values will be used for start and end of effort
`dropZeroes`	logical flag to remove boxplots where all observations are zero (these would normally appear as a flat line at zero)
`title`	optional title for the plot
`returnData`	if `TRUE` then no plot will be generated, instead the dataframe that would normally be used to make the plot will be returned

Details

The combination of `group`, `facet`, and `combineYears` determine the data points that make up each boxplot. If `combineYears=TRUE`, then there will be a different point for each year. There will additionally be separate points for each different value of the columns in `group`, excluding the column used for `facet` (since these points are instead split out to different facetted plots).

For example, if you have data from a single location, then settings of `combineYears=FALSE`, `group='species'`, and `facet=NULL` will create a plot where each point in a boxplot represents the number of detections for a species. If you change to `facet='species'`, then the result will show a multi panel plot where each boxplot is just a single point. Then changing to `combineYears=TRUE` will show a multi panel plot where each point in a boxplot is the number of detections for that panel's species in different years.

Value

a ggplot object

Author(s)

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

`plotHourlyLevel` *Plot Hourly Sound Level*

Description

Plots a heatmap of summarised sound levels. Y-axis is hour of the day, X-axis is frequency bin. Plotted values are the median of the `value` column for each hour/frequency pairing across the dataset. This function is designed to work with sound level outputs with consistent frequency bins measured across time

Usage

```
plotHourlyLevel(
  x,
  title = NULL,
  units = NULL,
  scale = c("log", "linear"),
  freqMin = NULL,
  dbRange = NULL,
  toTz = "UTC",
  cmap = viridis_pal()(25),
  returnData = FALSE
)
```

Arguments

`x`	a dataframe with columns `UTC`, `frequency`, and `value`
`title`	title for the plot. If `NULL` (default) it will use the first value in the `type` column of `x` (if present)
`units`	name of units for plot labeling, default is taken from common soundscape units
`scale`	one of `'log'` or `'linear'` for the scale of the frequency axis
`freqMin`	minimum frequency for the plot range, if desired to be different than the minimum frequency of the data
`dbRange`	range of dB values to plot
`toTz`	timezone to use for the time axis (input data must be UTC). Specification must be from OlsonNames
`cmap`	color palette map to use for plot, default is viridis_pal
`returnData`	if `TRUE` then no plot will be generated, instead the dataframe that would normally be used to make the plot will be returned

Value

a ggplot object

Author(s)

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

Examples

```
plotHourlyLevel(system.file('extdata/OLSmall.csv', package='PAMscapes'))
```

`plotLTSA` *Plot Long-Term Spectral Average (LTSA)*

Description

Creates a long-term spectral average (LTSA) style plot of the data, a plot where the x-axis is time and the y-axis is frequency. Color represents the magnitude of sound. In order to compress the time axis, data are binned into time chunks and the median value within that time bin is displayed

Usage

```
plotLTSA(
  x,
  bin = "1hour",
  scale = c("log", "linear"),
  title = NULL,
  freqRange = NULL,
  dbRange = NULL,
  units = NULL,
  facet = NULL,
  cmap = viridis_pal()(25),
  toTz = "UTC",
  alpha = 1,
  maxBins = 800,
  returnData = FALSE
)
```

Arguments

`x`	a soundscape metric file that can be read in with loadSoundscapeData, or a dataframe with `UTC`, `frequency`, and `value`
`bin`	amount of time to bin for each LTSA slice, format can be "#Unit" e.g. `'2hour'` or `'1day'`
`scale`	scaling for frequency axis, one of `log` or `linear`
`title`	optional title for plot
`freqRange`	if not `NULL`, a vector of two numbers specifying the range of frequencies (Hz) to plot. Providing `NA` for either value will use the max/min frequency present in the dataset
`dbRange`	if not `NULL`, a fixed limit to use for the color scaling of dB values in the plot
`units`	units for plot labeling, will attempt to read them from the input
`facet`	optional column to facet by to create multiple LTSA plots in separate rows
`cmap`	color palette map to use for plot, default is viridis_pal
`toTz`	timezone to use for the time axis (input data must be UTC). Specification must be from OlsonNames
`alpha`	alpha to use for the plot fill
`maxBins`	the maximum number of time bins to create for the plot. If `bin` would divide the range of dates in `x` into more than `maxBins`, then a warning will be given and a larger time bin will be used that reduces the number of time bins plotted. Trying to show a large number of bins will cause this function to be much slower
`returnData`	if `TRUE` then no plot will be generated, instead the dataframe that would normally be used to make the plot will be returned

Value

ggplot object of the LTSA plot

Author(s)

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

Examples

```
hmd <- loadSoundscapeData(system.file('extdata/MANTAExampleSmall1.csv', package='PAMscapes'))
# time range is too small for nice plots
plotLTSA(hmd, bin='1min', title='Every Minute')
plotLTSA(hmd, bin='2min', title='2 Minute Bins')
```

`plotPolarDetections` *Plot Detections in a Polar Plot*

Description

Plots detection data in a polar plot where the circular (angular) axis is either the hour of day or month of the year

Usage

```
plotPolarDetections(
  x,
  bin = c("days/month"),
  quantity = c("count", "mean", "effort", "percentEffort", "percentTotal"),
  group = "species",
  facet = NULL,
  effort = NULL,
  title = NULL
)
```

Arguments

`x`	dataframe of data loaded with loadDetectionData
`bin`	character specification of the radial and circular (angular) dimensions of the plot in the form "bin1/bin2", where "bin1" is one of "detection", "hour", or "day", and "bin2" is one of "hour" or "month". "bin1" is the units of the radial axis, and "bin2" is the unit of the circular axis. If "bin1" is "detection", then each row is treated as a distinct instantaneous detection, otherwise calls are binned using binDetectionData.
`quantity`	character indicating what type of quantity to plot. "count" plots total detections, "mean" plots average detections across groups, "effort" plots amount of total effort, "percentTotal" plots number of detections as a percent of total detections, "percentEffort" plots percent of total effort with detections.
`group`	a vector of name(s) of columns in `x` indicating which rows are distinct from eachother, used for binning data and accounting for effort. Typically something like "species", "site", or both
`facet`	optional name of column to facet output plot by

`effort`	Not relevant for `quantity` "count or "percentTotal", or `bin` with "detection". If not `NULL`, a dataframe describing on effort times to be formatted with formatEffort. If effort data is not provided then times with zero detections will not be properly accounted for. Alternatively, if columns "effortStart" and "effortEnd" are present in `x`, then these values will be used for start and end of effort
`title`	optional title for plot

Value

a ggplot object

Author(s)

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

`plotPSD` *Plot Power Spectral Density*

Description

Plots the distribution of summarised sound levels across frequency, either as lines of quantile levels or a heatmap showing the full distribution. Multiple PSD sources can be combined and plotted as long as they have identical frequency levels.

Usage

```
plotPSD(
  x,
  style = c("quantile", "density"),
  scale = c("log", "linear"),
  q = 0.5,
  color = "black",
  freqRange = NULL,
  dbRange = NULL,
  dbInt = 1,
  densityRange = NULL,
  units = "dB re: 1uPa^2/Hz",
  cmap = viridis_pal()(25),
  by = NULL,
  referenceLevel = NULL,
  facet = NULL,
  ncol = NULL,
  title = NULL,
  returnData = FALSE,
  progress = TRUE
)
```

```
prepPSDData(
  x,
  freqRange = NULL,
  style = c("density", "quantile"),
  by = NULL,
  dbInt = 1,
  compression = 10000,
  progress = TRUE
)
```

Arguments

`x`	a dataframe or file path or vector of file paths
`style`	character specifying plot style to create, either "quantile", "density", or a vector with both
`scale`	scale to use for frequency axis, one of "log" or "linear"
`q`	quantile to plot
`color`	color for quantile
`freqRange`	range of frequencies to plot
`dbRange`	range of dB values to plot
`dbInt`	bin interval size for density plot
`densityRange`	optional range of values for density color scale
`units`	units for dB axis of plot
`cmap`	color map to use for density plot
`by`	optional column to plot different quantile lines by, only affects `style='quantile'`. If `x` is a data.frame, `by` can also be one of `'hour'`, `'month'`, or `'year'` and that column will be created automatically if not present.
`referenceLevel`	only used together with `by`. A value of the `by` column to use as a reference for all other levels. The plot will then show the difference between the other levels and the reference
`facet`	optional column to facet the plots by
`ncol`	number of columns to use when plotting with `facet`
`title`	optional title for plot
`returnData`	if `TRUE` then no plot will be generated, instead the dataframe that would normally be used to make the plot will be returned
`progress`	logical flag to show progress bar
`compression`	no longer used

Value

a ggplot object

Author(s)

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

Examples

```
psd <- loadSoundscapeData(system.file('extdata/PSDSmall.csv', package='PAMscapes'))
# Plotting only first 1000 columns for brevity
plotPSD(psd[1:1000], style='density')
plotPSD(psd[1:1000], style='quantile', q=.05)
```

`plotQAQCLevel` *Plot QAQC Data*

Description

Simple plotting functions for various the various types of quality assurance / quality control (QAQC) data created by evaluateDeployment and evaluateRecordings

Usage

```
plotQAQCLevel(
  x,
  level = c("ol", "tol"),
  dbRange = NULL,
  freqMin = NULL,
  title = NULL
)

plotQAQCGap(x, title = NULL)

plotQAQCTV(x, title = NULL)
```

Arguments

`x`	dataframe created by evaluateDeployment or evaluateRecordings
`level`	which frequency band levels to display, one of "ol" (octave level) or "tol" (third-octave level).
`dbRange`	range of dB (y-axis) values to plot
`freqMin`	minimum frequency (Hz) to show on plot
`title`	optional title to add to plot

Value

a ggplot object

`plotScaledTimeseries` *Plot Rescaled Timeseries*

Description

Plot timeseries of different values, rescaled so that multiple types of data are visible on the same plot

Usage

```
plotScaledTimeseries(
  x,
  columns,
  title = NULL,
  units = NULL,
  color = hue_pal(),
  cpal,
  lwd = 0.5,
  minVals = NA,
  relMax = 1,
  toTz = "UTC"
)
```

Arguments

`x`	a dataframe with column `UTC`
`columns`	the names of the columns to plot. Values of columns will be rescaled to appear similar to range of the first column
`title`	title for the plot
`units`	name of units for plot labeling, default is taken from common soundscape units
`color`	colors to use for different lines, can either be a color palette function or a vector of color names
`cpal`	Deprecated in favor of `color` for naming consistency
`lwd`	line width, either a single value or a vector of widths matching the length of `columns`
`minVals`	minimum value for each of `columns` to use for rescaling, either a single value to use for all or a vector matching the length of `columns`. A value of `NA` will use the minimum value present in the data. See Details for more info
`relMax`	the percentage of the maximum value for all rescaled columns relative to the first column. See Details for more info
`toTz`	timezone to use for the time axis (input data must be UTC). Specification must be from OlsonNames

Details

The data in the different `columns` of `x` may have very different ranges, so they must be rescaled in order to create a useful comparison plot. The default behavior is to rescale all columns to have the same min/max range as the first column in `columns`. This means that the Y-axis values will only be accurate for the first column, and all lines will have their minimum value at the bottom edge of the plot and their maximum value at the top edge of the plot.

There are some cases where this full-range rescaling is not desirable. One case is when one of the variables should have a minimum value of zero, but the lowest value present in your data is larger than zero. For example, wind speed might in your data might range from values of 0.5 to 3, so by default this 0.5 value would appear at the bottom of the plot. However, it would make much more sense if the values were plotted relative to a minimum of zero. The `minVals` argument lets you control this. The default `NA` value uses the minimum of your data range, but you can provide a value of zero (or anything else) to control the displayed minimum.

It can also be distracting or busy to display all lines at the same relative height, especially as the number of columns displayed grows. There are two ways to help this. First, the `lwd` parameter can be used to display certain lines more prominently, making it easier to keep track of more important information. Second, the `relMax` can be used to control the maximum relative height of each line plot. The default value of 1 makes each line the same maximum height as the first column, reducing this to a value of 0.75 would make it so that all lines other than the first will not go higher than 75% of the Y-axis

Value

a ggplot object

Author(s)

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

Examples

```
manta <- loadSoundscapeData(system.file('extdata/MANTAExampleSmall1.csv', package='PAMscapes'))
plotScaledTimeseries(manta, columns=c('HMD_50', 'HMD_100', 'HMD_200'))
```

`plotTimeseries` *Plot Timeseries*

Description

Plot simple timeseries of values

Usage

```
plotTimeseries(
  x,
  bin = "1hour",
  column,
  title = NULL,
  units = NULL,
  style = c("line", "heatmap"),
  q = 0,
  by = NULL,
  cmap = viridis_pal()(25),
  toTz = "UTC"
)
```

Arguments

`x`	a dataframe with column `UTC`
`bin`	time bin for summarising data. The median of values within the same time bin will be plotted
`column`	the name of the column to plot
`title`	title for the plot, if left as default `NULL` it will use the `column` name
`units`	name of units for plot labeling, default is taken from common soundscape units
`style`	one of `'line'` or `'heatmap'`. `'line'` will create a simple line time series plot, `'heatmap'` will create a grid plot with hour of day as X-axis and Date as y-axis where the value of `column` is the color
`q`	only valid for `style='line'`, quantile level for plotting, between 0 and 1. If left as `0`, none will be plotted. If a single value, then levels `q` and `1-q` will be plotted. Users can also specify both values for non-symmettric intervals.
`by`	only valid for `style='line'`, optional categorical column to plot separate lines for
`cmap`	only valid for `style='heatmap'`, the color palette to use for plotting values
`toTz`	timezone to use for the time axis (input data must be UTC). Specification must be from OlsonNames

Value

a ggplot object

Author(s)

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

Examples

```
manta <- loadSoundscapeData(system.file('extdata/MANTAExampleSmall1.csv', package='PAMscapes'))
plotTimeseries(manta, bin='1minute', column='HMD_150')
```

`readLocalAIS` *Read AIS Data Near GPS Track*

Description

Reads in AIS data downloaded from Marine Cadastre of ship tracks that come within a certain distance of a given GPS track. Also calculates the distance to the GPS track for each AIS point

Usage

```
readLocalAIS(gps, aisDir, distance = 10000, timeBuff = 0)
```

Arguments

`gps`	a dataframe with columns `UTC`, `Latitude`, and `Longitude` to get nearby AIS data for
`aisDir`	directory of AIS CSV files to read from
`distance`	distance in meters around the GPS track to read AIS data for
`timeBuff`	extra time (seconds) before and after the GPS points to read AIS data for. This can help create a better picture of ship activity surrounding the GPS

Value

a dataframe of AIS data, with additional columns related to distance to provided buoy GPS track

Author(s)

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

Examples

```
gps <- data.frame(Latitude=c(33.2, 33.5,33.6),
                  Longitude=c(-118.1, -118.4, -119),
                  UTC=as.POSIXct(
                    c('2022-04-28 05:00:00',
                      '2022-04-28 10:00:00',
                      '2022-04-28 20:00:00'),
                    tz='UTC'))
ais <- readLocalAIS(gps, aisDir=system.file('extdata/ais', package='PAMscapes'), distance=20e3)
str(ais)
```

`runDailyLTSAReview` *Run Daily Soundscape LTSA Data Review Shiny App*

Description

Run Daily Soundscape LTSA Data Review Shiny App

Usage

```
runDailyLTSAReview(file, plotQuality = FALSE)
```

Arguments

`file`	either a folder containing NetCDF soundscape files or a single file
`plotQuality`	logical flag whether or not to plot the matrix of data quality flags. This greatly increases the load time for each file.

Details

Launches a Shiny app that allows users to manually review LTSA style plots of soundscape NetCDF data. Users can draw annotations on the LTSA plot and export these annotations as a CSV file. Users can additionally mark selected areas with the 1-4 standard quality flags.

Value

nothing, just runs the app

Author(s)

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

`runDetectionExplorer` *Run Detection Data Explorer App*

Description

Runs a Shiny app that allows users to interactively find the proper parameters to load a detection dataset using loadDetectionData and explore plots for detection data

Usage

```
runDetectionExplorer(data = NULL)
```

Arguments

`data`	file path to a CSV file containing detection data or a dataframe

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
if(interactive()) {
  detFile <- system.file('extdata/detectionExample.csv', package='PAMscapes')
  runDetectionExplorer(detFile)
}
```

---

runDetLoadHelper *Run Detection Data Loading Helper App*

Description

Runs a Shiny app that allows users to interactively find the proper parameters to load a detection dataset using loadDetectionData. As of 0.14.0

Usage

```
runDetLoadHelper(data = NULL)
```

Arguments

data	file path to a CSV file containing detection data

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
if(interactive()) {
  detFile <- system.file('extdata/detectionExample.csv', package='PAMscapes')
  runDetLoadHelper(detFile)
}
```

`runQAQCReview` *Run QAQC Review Shiny App*

Description

Launches a shiny app that allows users to review QAQC outputs created by evaluateDeployment and interactively mark potential problems to investigate. Potential problems can be saved in an "Issues Log" that can be downloaded as a CSV file.

Usage

```
runQAQCReview(data, issue = NULL, freqLims = c(30, Inf))
```

Arguments

`data`	output from evaluateDeployment either as a dataframe or path to a CSV file containing the same data
`issue`	if not `NULL`, a dataframe or path to CSV file of issue logs previously created with this function
`freqLims`	frequency limits (Hz) for plotting

Value

invisibly `data`

Author(s)

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

`runSoundscapeExplorer` *Run Soundscape Explorer App*

Description

Launches a shiny app that allows users to browse the various plotting functions available to visualize soundscape data

Usage

```
runSoundscapeExplorer(data = NULL)
```

Arguments

`data`	file path to soundscape data or data that has been loaded with loadSoundscape-Data

Value

invisible TRUE

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
if(interactive()) {
 hmd <- loadSoundscapeData(system.file('extdata/MANTAExampleSmall1.csv', package='PAMscapes'))
  runSoundscapeExplorer(hmd)
}
```

`subsetMarCadAIS` *Subset Marine Cadastre AIS Data to Region*

Description

Subsets the full download files from Marine Cadastre to a smaller region so that they are easier to work with

Usage

```
subsetMarCadAIS(
  inDir,
  outDir,
  latRange = c(20, 50),
  lonRange = c(-140, -110),
  name = "West_",
  overwrite = FALSE,
  progress = TRUE
)
```

Arguments

`inDir`	directory containing Marine Cadastre AIS CSV files to subset
`outDir`	directory to write subsetted files to
`latRange`	range of desired latitudes (decimal degrees)
`lonRange`	range of desired longitudes (decimal degrees)
`name`	prefix to append to new filenames
`overwrite`	logical flag to overwrite existing files
`progress`	logical flag to show progress bar

Value

invisibly return new file names

Author(s)

Taiki Sakai <taiki.sakai@noaa.gov>

Examples

```
outDir <- tempdir()
localFiles <- subsetMarCadAIS('AISData', outDir=outDir,
                              latRange=c(20, 50), lonRange=c(-140, -110),
                              name='West_')
```

Index