Package 'locaR'

August 21, 2026

Type Package

Title A Set of Tools for Sound Localization

Version 0.3.0

Maintainer Richard Hedley <rwhedley@gmail.com>

Description A set of functions and tools to conduct acoustic source localization, as well as organize and check localization data and results. The localization functions implement the modified steered response power algorithm described by Cobos et al. (2011) <doi:10.1109/LSP.2010.2091502>.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/rhedley/locaR

BugReports https://github.com/rhedley/locaR/issues

RoxygenNote 7.3.3

Imports seewave, tuneR, matrixStats, oce, signal, SynchWave, stats, utils, graphics, grDevices

Suggests testthat (>= 2.0.0), knitr, rmarkdown

Config/testthat/edition 2

VignetteBuilder knitr

NeedsCompilation no

Author Richard Hedley [cre, aut] (ORCID: <https://orcid.org/0000-0001-9588-5515>), Marcus Becker [aut], Tim Huang [aut]

Repository CRAN

Date/Publication 2026-08-21 05:42:10 UTC

Contents

checkSettings . . . 2
createSettings . . . 3
createWavList . . . 5
getFilepaths . . . 6
layoutMatrix . . . 7
localize . . . 8
localizeSingle . . . 11
locHeatmap . . . 12
makeSearchMap . . . 14
MSRP_Init . . . 15
MSRP_RIJ_HT . . . 16
omniSpectro . . . 16
parseWAFileNames . . . 18
processSettings . . . 18
Rij_GCC . . . 19
setupSurvey . . . 20
spatialEntropy . . . 21
surveyPaths . . . 23
validationSpec . . . 24

Index **26**

`checkSettings` *Check the validity of a settings file or data.frame.*

Description

Several checks are run:

1. `settings` is either a valid file or a data.frame.
2. That the adjustments file is either an existing file or ""
3. That the channels file is either an existing file or NULL.
4. That the coordinates file exists.
5. That the detections file exists.
6. That the siteWavsFolder exists.
7. That buffer, margin, resolution, date, time, zMin, zMax and surveyLength can all be recognized as numbers.
8. That tempC or soundSpeed have been defined.

Usage

```
checkSettings(settings)
```

Arguments

settings	Character or data.frame. Either the path to a settingsFile (csv) or a data.frame containing settings.

Value

Logical, indicating whether all checks were passed or not.

---

createSettings *Create settings file (csv) or data frame by defining the localization settings.*

Description

createSettings takes a series of arguments and creates a data frame (or csv) with standard structure that can be read by other functions in the locaR package.

Usage

```
createSettings(
  projectName,
  run = 1,
  detectionsFile,
  coordinatesFile,
  siteWavsFolder,
  adjustmentsFile,
  channelsFile,
  date,
  time,
  tempC = 15,
  soundSpeed,
  surveyLength,
  margin = 10,
  zMin = -1,
  zMax = 10,
  resolution = 1,
  buffer = 0.2,
  write.csv = FALSE
)
```

Arguments

projectName	Character. A string specifying the name of a project.
run	Numeric. Within each survey, start with run = 1, then count upwards. So, run 2 would be used to re-localize sounds that were poorly localized in run 1, etc. Running them again with slightly different settings (e.g. start/end times or low/high frequencies) can improve results.

`detectionsFile` Character. File path to the detections file (csv).

`coordinatesFile` Character. File path to the coordinates file (csv).

`siteWavsFolder` Character. Folder path of the directory containing audio files. The folder path will be searched recursively if using `localizeSingle` or `localizeMultiple`.

`adjustmentsFile` Character. File path to the adjustments file (csv). Not required to be specified.

`channelsFile` Character. File path to the channels file (csv), specifying which channel (1 or 2) to use for each recording unit.

`date` Numeric. Eight digit number representing a date in the format YYYYMMDD.

`time` Numeric. Five or six digit number representing the start time of a recording session (90000 = 09:00:00, and 160000 = 16:00:00).

`tempC` Numeric. Temperature in degrees C, which is used to calculate the speed of sound in air using the equation 331.45*sqrt(1+tempC/273.15).

`soundSpeed` Numeric. The speed of sound in meters per second. If missing, the speed of sound is calculated based on the specified temperature (assuming the transmission medium is air). If soundSpeed is specified, the tempC value is over-ridden.

`surveyLength` Numeric. Length of the survey, in seconds.

`margin` distance (in meters) to extend the search grid beyond the x-y limits of the microphone locations. The same buffer is applied to x and y coordinates.

`zMin` distance (in meters) to begin grid search relative to the microphone with the lowest elevation. Typically a small negative number to ensure that the grid search begins slightly below the lowest microphone.

`zMax` distance (in meters) to end search relative to the microphone with the highest elevation. Typically a positive number to ensure that the grid search ends well above the highest microphone.

`resolution` resolution of the search map, in meters.

`buffer` Amount of time (in seconds) to expand each detection. This accounts for imprecise time boundaries, and also the differences imposed by time delays between different microphones (e.g. two microphones separated by some amount will receive the same sound at different times).

`write.csv` Logical. Whether or not to write a settingsFile csv. The csv will be written to the same directory as the detections file.

Value

A data frame with columns Setting and Value specifying the value of each setting needed for localization.

createWavList *Create a list of Wave objects.*

Description

This function reads in portions of a set of synchronized .wav files. It is intended to be used to load sounds of interest for localization.

Usage

```
createWavList(
  paths,
  names,
  from,
  to,
  buffer,
  adjustments,
  channels,
  index = "unknown"
)
```

Arguments

paths	Character vector. File paths to the set of .wav files to be read.
names	Character vector. Station names for the files. Must have the same length and the names must occur in the same order as the paths variable.
from	Numeric. Start time, in seconds, of the sound of interest, relative to the start of the file.
to	Numeric. End time, in seconds, of the sound of interest, relative to the start of the file.
buffer	Numeric. Amount of blank space around each sound of interest to be read.
adjustments	Numeric vector. Amount, in seconds, to adjust the start times of recordings, if not already synchronized. Vector must be of the same length as the paths variable. If not specified, default is no adjustment.
channels	Numeric vector. The channel to be read from each .wav file. Left = 1, Right = 2. If missing, default is left channel (channel 1) for all recordings.
index	Numeric. If using this function within a loop, pass the index i to the function, which can help with troubleshooting if an error occurs.

Value

Named list of Wave objects.

Examples

```
#list example mp3 files.
wavs <- list.files(system.file('extdata', package = 'locaR'),
                   pattern = 'mp3$', full.names = TRUE)
#get names of mp3 locations.
nms <- substr(basename(wavs), 1, 4)
#create wave list.
wl <- createWavList(paths = wavs, names = nms, from = 1, to = 2, buffer = 0.1)
```

`getFilepaths` *Get filepath information for a date and time.*

Description

`getFilepaths` reads information from a settings file (csv) or a settings list and returns the file paths and other information as a dataframe. It undertakes a recursive search within the site folder for files matching the date and time.

Usage

```
getFilepaths(settings, types = "wav")
```

Arguments

`settings`	Either a filepath to a settings file (csv) or a settings list. If a filepath, the filepath will first be passed to `processSettings`.
`types`	Character, specifying the file type to be searched for. Either 'wav' or 'mp3'.

Value

A data frame with station names, coordinates, filepaths, and any recording start-time adjustments.

Examples

```
#Read example data
settings <- read.csv(system.file('extdata',
               'Ex_20200617_090000_Settings.csv', package = 'locaR'),
               stringsAsFactors = FALSE)

#Over-write default values for SiteWavsFolder, CoordinatesFile, and ChannelsFile
settings$Value[settings$Setting == 'SiteWavsFolder'] <-
              system.file('extdata', package = 'locaR')
settings$Value[settings$Setting == 'CoordinatesFile'] <-
               system.file('extdata', 'Vignette_Coordinates.csv',
                           package = 'locaR')
settings$Value[settings$Setting == 'ChannelsFile'] <-
               system.file('extdata', 'Vignette_Channels.csv',
                           package = 'locaR')
```

```
#Run processSettings() function
st <- processSettings(settings = settings, getFilepaths = FALSE)

#Get filepaths.
fp <- getFilepaths(settings = st, types = 'mp3')
```

`layoutMatrix` *Specify the spatial layout of microphones.*

Description

`layoutMatrix` creates a matrix of station names, which correspond to the layout of stations in space.This is passed to the `omniSpectro` function for the purposes of generating spectrograms that align with the spatial orientation of stations. The four user-specified arguments indicate where the first station occurs (e.g. topleft means the first station is in the northwest; the "first" station means the one with the name that would appear first when sorted alphabetically). byrow means the stations increase along rows (either left to right or right to left) and nrow and ncol indicate how many rows and columns of microphones there are (assuming the array has a rectangular shape). Note that this layout function is provided for convenience, but users can easily specify their own custom layouts manually.

Usage

```
layoutMatrix(
  st,
  stationNames = NULL,
  start = c("topleft", "topright", "bottomleft", "bottomright"),
  byrow = TRUE,
  nrow,
  ncol
)
```

Arguments

`st`	List. Localization settings object generated using `processSettings`.
`stationNames`	Character vector. Vector of station names. Not required if st is provided.
`start`	Character. When sorted alphabetically, the location of the first station name.
`byrow`	Logical. An indicator of whether station names increase along rows (TRUE) or along columns (FALSE)
`nrow`	Numeric. The number of rows of microphones in the layout.
`ncol`	Numeric. The number of columns of microphones in the layout.

Value

Matrix, containing the station names within the array. If done correctly, the matrix rows and columns should align with the spatial layout of the stations in the field.

Examples

```
#Vector of station names, Ex-1 to Ex-9.
stationNames <- paste0('Ex-',1:9)

#All options shown below.
#layoutMatrix starting from top left (NW) to bottom right (SE) by row.
layoutMatrix(stationNames = stationNames,
             start = 'topleft', byrow = TRUE, ncol = 3, nrow = 3)

#layoutMatrix starting from top left (NW) to bottom right (SE) by column.
layoutMatrix(stationNames = stationNames,
             start = 'topleft', byrow = FALSE, ncol = 3, nrow = 3)

#layoutMatrix starting from top right (NE) to bottom left (SW) by row.
layoutMatrix(stationNames = stationNames,
             start = 'topright', byrow = TRUE, ncol = 3, nrow = 3)

#layoutMatrix starting from top right (NE) to bottom left (SW) by column.
layoutMatrix(stationNames = stationNames,
             start = 'topright', byrow = FALSE, ncol = 3, nrow = 3)

#layoutMatrix starting from bottom left (SW) to top right (NE) by row.
layoutMatrix(stationNames = stationNames,
             start = 'bottomleft', byrow = TRUE, ncol = 3, nrow = 3)

#layoutMatrix starting from bottom left (SW) to top right (NE) by column.
layoutMatrix(stationNames = stationNames,
             start = 'bottomleft', byrow = FALSE, ncol = 3, nrow = 3)

#layoutMatrix starting from bottom right (SE) to top left (NW) by row.
layoutMatrix(stationNames = stationNames,
             start = 'bottomright', byrow = TRUE, ncol = 3, nrow = 3)

#layoutMatrix starting from bottom right (SE) to top left (NW) by column.
layoutMatrix(stationNames = stationNames,
             start = 'bottomright', byrow = FALSE, ncol = 3, nrow = 3)
```

`localize` *Localize detected sounds*

Description

`localize` and the related function `localizeMultiple` are the basic functions for localizing sounds. They take audio data as inputs, alongside relevant metadata (e.g. coordinates and a variety of settings), and estimate the location of the dominant sound source. The `localize` function takes as arguments the minimal amount of information needed for localization. Localization is conducted on the full duration of the Wave objects in wavList. Effectively this means the user must wrangle the data and clip the audio themselves, but this affords the greatest flexibility in terms of how the user chooses to organize their data. The `localizeMultiple` function, in contrast, automates much

of the data wrangling process, but requires data to be organized in a very specific way (e.g. folder structure, file structures). Thus, the latter function trades off flexibility for increased automation. Both functions use the same underlying localization algorithm - `localizeMultiple` passes its data to `localize` after the data has been wrangled.

Usage

```
localize(
  wavList,
  coordinates,
  margin = 10,
  zMin = -1,
  zMax = 20,
  resolution = 1,
  F_Low = 2000,
  F_High = 8000,
  tempC = 15,
  soundSpeed,
  plot = TRUE,
  locFolder,
  jpegName = "000.jpeg",
  InitData = NULL,
  keep.InitData = TRUE,
  keep.SearchMap = FALSE,
  plot_label
)

localizeMultiple(st, indices = "all", plot = TRUE, InitData = NULL)
```

Arguments

`wavList`	list of Wave objects. The name of the Wave objects MUST be present in the coordinates data.frame.
`coordinates`	data.frame. Must contain four required columns: column Station contains a character string with names of each recording station, while Easting, Northing and Elevation contain the x, y, and z coordinates of the station, in meters (E.g. UTM coordinates).
`margin`, `zMin`, `zMax`, `resolution`	Arguments describing the area to be searched for sound sources. Passed to `makeSearchMap`.
`F_Low`, `F_High`	Numeric. The low and high frequency, in Hz, of the sound to be localized.
`tempC`	Numeric. Temperature in degrees C, which is used to calculate the speed of sound in air using the equation 331.45*sqrt(1+tempC/273.15).
`soundSpeed`	Numeric. The speed of sound in meters per second. If missing, the speed of sound is calculated based on the specified temperature (assuming the transmission medium is air). If soundSpeed is specified, the tempC value is over-ridden.
`plot`	Logical. Whether to plot jpegs.

locFolder	Character. File path to the folder where localization jpegs (heatmaps and spectrograms) are to be created. Only required if plot = TRUE.
jpegName	Character. Name of the jpeg, ending in extension .jpeg. Only required if plot = TRUE.
InitData	List. An InitData list created by running localization with keep.InitData = TRUE. Providing an InitData list saves computation time, but is only possible if the SearchGrid and stations used for localization remain unchanged. Default is NULL, which means the InitData will be calculated anew.
keep.InitData	Logical. Whether to store the InitData list.
keep.SearchMap	Logical. Whether to keep the SearchMap list with power estimates and coordinates of each grid cell. Should only be set to TRUE if the SearchMap is needed for some other reason (e.g. making a publication-ready figure or conducting more involved analysis with overlapping sources, etc.).
plot_label	Character. If provided, and plot=TRUE, text that will be added to the bottom-right of the validation plots. Can be useful, for example, to keep track of which song is being localized.
st	List. Localization settings object generated using processSettings. Only needed for localizeSingle or localizeMultiple.
indices	Numeric or 'all'. Indices to be localized within a detection file. Setting to 1 localizes the first row, c(7:10) localizes rows 7-10, and 'all' localizes all rows (ignoring rows that have no entry in the Station1 column).

Value

List, containing the location of the sound source (global maximum), and optionally the InitData and SearchMap lists.

References

Cobos, M., Martí, A., & J.J. López. 2011. A modified SRP-PHAT functional for robust real-time sound source localization with scalable spatial sampling. IEEE Signal Processing Letters. 18:71-74. doi:10.1109/LSP.2010.2091502.

Examples

```
## example for localize()
#Get filepaths for example data.
fp <- list.files(system.file('extdata', package = 'locaR'), pattern = '.mp3', full.names = TRUE)
#Add names.
names(fp) <- sapply(strsplit(basename(fp), '_'), '[[', 1)
#Load first row of detection data.
row <- read.csv(system.file('extdata',
     'Vignette_Detections_20200617_090000.csv', package = 'locaR'),
      stringsAsFactors = FALSE)[1,]
#Get non-empty Station columns.
stationSubset <- unlist(row[1,paste0('Station',1:6)])
stationSubset <- stationSubset[!is.na(stationSubset) & stationSubset != '']
#Create wav list.
```

```
wl <- createWavList(paths = fp[stationSubset], names = stationSubset,
       from = row$From, to = row$To, buffer = 0.2, index=1)
#Read coordinates.
coordinates <- read.csv(system.file('extdata', 'Vignette_Coordinates.csv',
                                    package = 'locaR'), stringsAsFactors = FALSE)
row.names(coordinates) <- coordinates$Station
#Subset coordinates.
crd <- coordinates[stationSubset,]
#Localize.
loc <- localize(wavList = wl, coordinates = crd, locFolder = tempdir(),
           F_Low = row$F_Low, F_High = row$F_High,
           jpegName = '0001.jpeg', keep.SearchMap = TRUE)

## Example for localizeMultiple().
#list mp3 files.
f.in <- list.files(system.file('extdata', package = 'locaR'), full.names = TRUE, pattern='mp3$')
#create wav names.
f.out <- file.path(tempdir(), basename(f.in))
#change extension.
substr(f.out, nchar(f.out)-2, nchar(f.out)) <- 'wav'
#Convert mp3 to wav, as required for this particular example.
for(i in 1:length(f.in)) {
  y <- tuneR::readMP3(f.in[i])
  tuneR::writeWave(y, filename = f.out[i])
}
#Set up survey.
survey <- setupSurvey(folder = tempdir(), projectName = 'Ex', run = 1,
           coordinatesFile = system.file('extdata', 'Vignette_Coordinates.csv',
                                         package = 'locaR'),
      siteWavsFolder = tempdir(), date = '20200617', time = '090000', surveyLength = 7)
#read example detections.
dets <- read.csv(system.file('extdata', 'Vignette_Detections_20200617_090000.csv',
                             package = 'locaR'))
#over-write empty detections file.
write.csv(dets, file.path(tempdir(), '20200617_090000',
          'Run1', 'Ex_20200617_090000_Run1_Detections.csv'), row.names = FALSE)
#Process settings.
st <- processSettings(settings = survey, getFilepaths = TRUE, types = 'wav')
#localize
locs <- localizeMultiple(st = st, indices = 1:2)
```

`localizeSingle` *Localize detected sounds*

Description

`localizeSingle` is an internal function implemented within `localizeMultipe`. Its basic function is to take an index value corresponding to a detection, extract that detection, extract the relevant coordinates, and feed all relevant metadata into the 'localize()' function.

Usage

```
localizeSingle(
  st,
  index,
  plot = TRUE,
  InitData = NULL,
  keep.InitData = TRUE,
  keep.SearchMap = FALSE
)
```

Arguments

st	List. Localization settings object generated using processSettings.
index	Numeric. Index to be localized within a detection file.
plot	Logical. Whether to plot jpegs.
InitData	List. An InitData list created by running localization with keep.InitData = TRUE. Providing an InitData list saves computation time, but is only possible if the SearchGrid and stations used for localization remain unchanged. Default is NULL, which means the InitData will be calculated anew.
keep.InitData	Logical. Whether to store the InitData list.
keep.SearchMap	Logical. Whether to keep the SearchMap list with power estimates and coordinates of each grid cell. Should only be set to TRUE if the SearchMap is needed for some other reason (e.g. making a publication-ready figure or conducting more involved analysis with overlapping sources, etc.).

Value

List, containing the location of the sound source (global maximum), and optionally the InitData and SearchMap lists.

---

locHeatmap *Create a heatmap to visualize localization output.*

Description

This function can be used to create a heatmap from the localization grid search. In general, this function should only be used internally, but it could be useful for making customized figures.

Usage

```
locHeatmap(SearchMap, SMap, NodeInfo, location, mar)
```

Arguments

SearchMap	An array created by the localize() function containing x, y and z coordinates. Created by setting keep.SearchMap = TRUE when running the localize() function.
SMap	An array created by the localize() function containing the power values. Created by setting keep.SearchMap = TRUE when running the localize() function.
NodeInfo	A list with two elements. First element Num is numeric, specifying the number of microphones used for localization. Second element Pos is a matrix of coordinates with column names Easting, Northing and Elevation, and row names corresponding to the Station (i.e. location) names.
location	Data frame. The location estimate of the sound source. Four columns: Easting, Northing, Elevation, Power. Data frame should only contain one row.
mar	Numeric vector with four elements. Passed to `oce::imagep()` for plotting.

Value

No return value.

Examples

```
 #Get filepaths for example data.
fp <- list.files(system.file('extdata', package = 'locaR'), pattern = '.mp3', full.names = TRUE)
 #Add names.
 names(fp) <- sapply(strsplit(basename(fp), '_'), '[[', 1)
 #Load first row of detection data.
 row <- read.csv(system.file('extdata',
      'Vignette_Detections_20200617_090000.csv', package = 'locaR'),
       stringsAsFactors = FALSE)[1,]
 #Get non-empty Station columns.
 stationSubset <- unlist(row[1,paste0('Station',1:6)])
 stationSubset <- stationSubset[!is.na(stationSubset) & stationSubset != '']
 #Create wav list.
 wl <- createWavList(paths = fp[stationSubset], names = stationSubset,
        from = row$From, to = row$To, buffer = 0.2, index=1)
 #Read coordinates.
 coordinates <- read.csv(system.file('extdata',
                         'Vignette_Coordinates.csv', package = 'locaR'),
                          stringsAsFactors = FALSE)
 row.names(coordinates) <- coordinates$Station
 #Subset coordinates.
 crd <- coordinates[stationSubset,]
 #Localize.
 loc <- localize(wavList = wl, coordinates = crd, locFolder = tempdir(),
            F_Low = row$F_Low, F_High = row$F_High,
            jpegName = '0001.jpeg', keep.SearchMap = TRUE)
 #Convert crd (coordinates) to matrix called NodePos.
 NodePos <- as.matrix(crd[,c('Easting', 'Northing', 'Elevation')])
 colnames(NodePos) <- c('Easting', 'Northing', 'Elevation')
 row.names(NodePos) <- crd$Station
 #Plot heatmap with locHeatmap().
```

```
locHeatmap(SearchMap = loc$SearchMap, SMap = loc$SMap,
           NodeInfo = list(Num = 5, Pos = NodePos), location = loc$location,
           mar = c(0,0,0,0))
```

`makeSearchMap` *Create a grid over which to search for sound sources.*

Description

`makeSearchMap` creates the three-dimensional array over which to search for sound sources.

Usage

```
makeSearchMap(
  easting,
  northing,
  elevation,
  margin = 10,
  zMin = -1,
  zMax = 10,
  resolution = 1
)
```

Arguments

`easting`	vector of x coordinates of microphones.
`northing`	vector of y coordinates of microphones.
`elevation`	vector of z coordinates of microphones.
`margin`	distance (in meters) to extend the search grid beyond the x-y limits of the microphone locations. The same buffer is applied to x and y coordinates.
`zMin`	distance (in meters) to begin search relative to the microphone with the lowest elevation. Typically a small negative number to ensure that the grid search begins slightly below the lowest microphone.
`zMax`	distance (in meters) to end search relative to the microphone with the highest elevation. Typically a positive number to ensure that the grid search ends well above the highest microphone.
`resolution`	resolution of the search map, in meters.

Details

The localization algorithms used in this package can search for sound sources over areas with arbitrary size and with arbitrary resolution. However, speed can sometimes be slow. Generally speaking, the speed of localization calculations correlates directly with the number of grid cells to be

searched. Speed can therefore be increased by searching a smaller area (i.e. by reducing the margin, increasing zMin, or decreasing zMax), or by searching with a coarser grain (i.e. by increasing the resolution).

The final list defining the search map includes three arrays containing x, y and z coordinates of each grid cell, as well as the resolution and range of values in the x, y and z directions. This list is passed to other functions for localization.

Value

A list defining the search map.

Examples

```
 #read coordinates.
coords <- read.csv(system.file('extdata', 'Vignette_Coordinates.csv', package = 'locaR'),
    stringsAsFactors = FALSE)
 #make search map.
 sm <- makeSearchMap(easting = coords$Easting,
                     northing = coords$Northing,
                     elevation = coords$Elevation)
```

`MSRP_Init` *Create InitData.*

Description

Internal function which creates the InitData list.

Usage

```
MSRP_Init(NodeInfo, SearchMap, Para, LevelFlag)
```

Arguments

`NodeInfo`	List with elements Num, Pos.
`SearchMap`	List with elements XDen, YDen, ZDen, XMap, YMap, ZMap
`Para`	List with Fs, Vc (speed of sound), and DataLen
`LevelFlag`	Integer. Only value currently supported is 2.

Value

List.

Author(s)

Tim Huang.

MSRP_RIJ_HT *Internal function for localization.*

Description

This function uses the InitData and other info to calculate the likelihood of sound sources coming from each location. Note: the LevelFlag argument is currently redundant because there is only one option. Similarly, the MSRP_HT_Level2 function could be rolled into the MSRP_RIJ_HT function in the future, but for now is kept separate.

Usage

```
MSRP_RIJ_HT(NodeInfo, SearchMap, Data, Para, LevelFlag, InitData)

MSRP_HT_Level2(NodeInfo, SearchMap, Data, Para, InitData)
```

Arguments

NodeInfo	List with elements Num, Pos.
SearchMap	List with elements XDen, YDen, ZDen, XMap, YMap, ZMap
Data	Matrix containing the wave samples.
Para	List with Fs, Vc (speed of sound), and DataLen
LevelFlag	Integer. Only value currently supported is 2.
InitData	List. Created with the MSRP_Init function.

Value

List.

Author(s)

Tim Huang.

omniSpectro *Generate grid of spectrograms for detecting sounds of interest.*

Description

omniSpectro creates a grid of time-synchronized spectrograms, to facilitate the manual detection of birds across a microphone array. By opening the resulting jpeg images in an image viewing program (e.g. the standard Microsoft Photos app), short clips of sounds can be viewed across an entire microphone array at once. The authors of this package have found this to be an efficient way to view spectrograms, while effectively eliminating the likelihood of double-counting sound sources that may be clearly detectable on many microphones at the same time. At the present time, this function only works when a settings object, st, is provided.

Usage

```
omniSpectro(st, lm, intervalLength = 5, intervals = "all")
```

Arguments

st	List. Localization settings object generated using processSettings.
lm	layout matrix generated using the 'layoutMatrix()' function, or a user-generated matrix in the same format. This matrix controls how the spectrograms from each station are mapped to rows and columns.
intervalLength	Integer The length of each view interval to be generated, in seconds. Consecutive windows overlap, by default by 1 second. Setting intervalLength = 5 will therefore create 6-second spectrogram views, with one second overlap (e.g. 0 to 6, then 5 to 11, 10 to 16, etc.).
intervals	Integer or 'all'. Which intervals to write to jpeg. For testing purposes, it is often desirable to set this to, e.g. intervals = 1:5, which will create only the first five view windows, to ensure the function is working.

Value

No return value.

Examples

```
#First need to convert mp3 example data to wav.
#list mp3 files.
f.in <- list.files(system.file('extdata', package = 'locaR'), full.names = TRUE, pattern='mp3$')
#create wav names.
f.out <- file.path(tempdir(), basename(f.in))
#change extension.
substr(f.out, nchar(f.out)-2, nchar(f.out)) <- 'wav'
#Convert mp3 to wav, as required for this particular example.
for(i in 1:length(f.in)) {
  y <- tuneR::readMP3(f.in[i])
  tuneR::writeWave(y, filename = f.out[i])
}
#Set up survey.
survey <- setupSurvey(folder = tempdir(), projectName = 'Ex', run = 1,
           coordinatesFile = system.file('extdata', 'Vignette_Coordinates.csv',
                                         package = 'locaR'),
      siteWavsFolder = tempdir(), date = '20200617', time = '090000', surveyLength = 7)
#Process settings.
st <- processSettings(settings = survey, getFilepaths = TRUE, types = 'wav')
#Set up layout matrix.
lm <- layoutMatrix(st = st, start = 'topleft', byrow = TRUE, nrow = 3, ncol = 3)
#create detection spectrograms.
omniSpectro(st, lm, intervalLength = 7)
```

parseWAFileNames *Parse Wildlife Acoustics-type file names.*

Description

This function parses the information in file names that are structured according to Wildlife Acoustics' naming convention. Specifically, the format prefix_date_time.wav or prefix_mic_date_time.wav.

Usage

```
parseWAFileNames(filenames)
```

Arguments

filenames	Character vector of file names.

Value

A data frame with prefix, channels, date, time and extension information.

processSettings *Process settings file to extract relevant information.*

Description

processSettings reads information from a settings file (csv) and combines them into a list for subsequent localization.

Usage

```
processSettings(settingsFile, settings, getFilepaths = FALSE, types = "wav")
```

Arguments

settingsFile	Filepath to the settings file (csv).
settings	data.frame created either by reading a settings file (csv) or using the createSettings function. Not needed if settingsFile is specified.
getFilepaths	Logical, indicating whether to add filepath information using getFilepaths.
types	Character. If getFilepaths is TRUE, which types of files to look for ('wav' or 'mp3').

Value

A list with information needed for sound localization, including microphone coordinates, the existing detections, channels to use for each recording unit, and information specifying the size and resolution of the grid within which to localize sound sources.

Examples

```
#Read example data
settings <- read.csv(system.file('extdata', 'Ex_20200617_090000_Settings.csv',
                                 package = 'locaR'), stringsAsFactors = FALSE)

#Over-write default values for SiteWavsFolder, CoordinatesFile, and ChannelsFile
settings$Value[settings$Setting == 'SiteWavsFolder'] <-
               system.file('extdata', package = 'locaR')
settings$Value[settings$Setting == 'CoordinatesFile'] <-
               system.file('extdata', 'Vignette_Coordinates.csv', package = 'locaR')
settings$Value[settings$Setting == 'ChannelsFile'] <-
               system.file('extdata', 'Vignette_Channels.csv', package = 'locaR')

#Run processSettings() function
st <- processSettings(settings = settings, getFilepaths = FALSE)
```

Rij_GCC *Generalized cross-correlation.*

Description

Internal function that calculates the generalized cross correlation.

Usage

```
Rij_GCC(data1, data2, Para)
```

Arguments

data1, data2	Wave samples.
Para	List with GCCMethod, FL, FH, Fs

Value

Numeric vector.

Author(s)

Tim Huang

setupSurvey *Set up a new "survey" with a standardized structure recognized by the package.*

Description

setupSurvey sets up the folder structure for a new "survey", which corresponds to a single recording session. By setting up a standardized folder structure, the package functions can carry out much of the data wrangling automatically using the localizeSingle and localizeMultiple functions. The extra work required to set up surveys in a standard format can save time later on.

Usage

```
setupSurvey(
  folder,
  projectName,
  run = 1,
  coordinatesFile,
  siteWavsFolder,
  adjustmentsFile,
  channelsFile,
  date,
  time,
  tempC = 15,
  soundSpeed,
  surveyLength,
  margin = 10,
  zMin = -1,
  zMax = 20,
  resolution = 1,
  buffer = 0.2
)
```

Arguments

folder	Character. Path to the directory where the survey will be created.
projectName, run, coordinatesFile, siteWavsFolder	Arguments passed to createSettings
adjustmentsFile	Character. File path to the adjustments file (csv). Optional argument.
channelsFile	Character. File path to the adjustments file (csv). If missing, an empty channels file (csv) will be created.
date	Numeric. Eight digit number representing a date in the format YYYYMMDD.
time	Numeric. Five or six digit number representing the start time of a recording session (90000 = 09:00:00, and 160000 = 16:00:00).

`tempC` Numeric. Temperature in degrees C, which is used to calculate the speed of sound in air using the equation 331.45*sqrt(1+tempC/273.15).

`soundSpeed` Numeric. The speed of sound in meters per second. If missing, the speed of sound is calculated based on the specified temperature (assuming the transmission medium is air). If soundSpeed is specified, the tempC value is over-ridden.

`surveyLength`, `margin`, `zMin`, `zMax`, `resolution`, `buffer`
Arguments describing the area to be searched for sound sources. Passed to `createSettings`.

Value

data.frame containing the settings generated using `createSettings`. This data.frame is identical to that produced by reading the settingsFile csv, which is also written to file.

Examples

```
survey <- setupSurvey(folder = tempdir(), projectName = 'Ex', run = 1,
 coordinatesFile = system.file('extdata', 'Vignette_Coordinates.csv', package = 'locaR'),
    siteWavsFolder = tempdir(),
    date = '20200617', time = '090000', surveyLength = 7)
```

`spatialEntropy` *Calculate spatial entropy.*

Description

This function returns a measure of spatial entropy within a user-specified distance of the estimated source location. The Power values from the location search are first restricted to the desired buffer_size distance (i.e. within buffer_size meters of the maximum value). The user can specify whether the buffer_size distance is calculated in three dimensions (a sphere) or two dimensions (a cylinder). Next, Power values are converted to a probability density function using the formula

$$p_{x,y,z} = \frac{\exp(\alpha \cdot power_{x,y,z})}{\sum_{x,y,z} \exp(\alpha \cdot power_{x,y,z})}$$

Where alpha is a tuning parameter. Normalized spatial entropy (Shannon's entropy) is then calculated using the probability density function values with the following formula

$$spatialEntropy = \frac{-\sum_{i=1}^{N}(p_i \cdot ln(p_i))}{ln(N)}$$

Where N is the number of pixels included within the restricted search area. By default, entropy is calculated within 10 meters of the location estimate, but other values may be desirable depending on the size and resolution of the array.

Usage

```
spatialEntropy(
  SearchMap,
  SMap,
  location,
  buffer_size = 10,
  method = c("cylinder", "sphere"),
  alpha = 5
)
```

Arguments

SearchMap	An array created by the localize() function containing x, y and z coordinates. Created by setting keep.SearchMap = TRUE when running the localize() function.
SMap	An array created by the localize() function containing the power values. Created by setting keep.SearchMap = TRUE when running the localize() function.
location	Data frame. The location estimate of the sound source. Four columns: Easting, Northing, Elevation, Power. Data frame should only contain one row.
buffer_size	Numeric. The distance from 'location' within which the spatial entropy should be calculated. If method == 'cylinder', all cells within this distance in the x-y direction are used. If method == 'sphere', distance is calculated in three dimensions.
method	Character. Either 'cylinder' or 'sphere' to extract power values based on two- or three-dimensional distances, respectively.
alpha	Numeric. Tuning parameter which scales the sensitivity when converting from power values to a probability density function. Smaller alpha values will push entropy values closer to 1, and larger alpha values will push entropy values towards zero.

Value

Numeric. Spatial entropy of the location estimate.

Examples

```
 #Get filepaths for example data.
fp <- list.files(system.file('extdata', package = 'locaR'), pattern = '.mp3', full.names = TRUE)
 #Add names.
 names(fp) <- sapply(strsplit(basename(fp), '_'), '[[', 1)
 #Load first row of detection data.
 row <- read.csv(system.file('extdata',
      'Vignette_Detections_20200617_090000.csv', package = 'locaR'),
       stringsAsFactors = FALSE)[1,]
 #Get non-empty Station columns.
 stationSubset <- unlist(row[1,paste0('Station',1:6)])
 stationSubset <- stationSubset[!is.na(stationSubset) & stationSubset != '']
 #Create wav list.
```

```
wl <- createWavList(paths = fp[stationSubset], names = stationSubset,
       from = row$From, to = row$To, buffer = 0.2, index=1)
#Read coordinates.
coordinates <- read.csv(system.file('extdata',
                        'Vignette_Coordinates.csv', package = 'locaR'),
                         stringsAsFactors = FALSE)
row.names(coordinates) <- coordinates$Station
#Subset coordinates.
crd <- coordinates[stationSubset,]
#Localize.
loc <- localize(wavList = wl, coordinates = crd, locFolder = tempdir(),
           F_Low = row$F_Low, F_High = row$F_High,
           jpegName = '0001.jpeg', keep.SearchMap = TRUE)

#Calculate spatial entropy.
spatialEntropy(SearchMap = loc$SearchMap, SMap = loc$SMap, location = loc$location,
      buffer_size = 10, method = 'cylinder', alpha = 5)
```

`surveyPaths` *Get paths for standardized survey workflow.*

Description

Function that takes arguments of a base folder and a project name, date, time, and run, and returns the appropriate filepaths for a standardized survey workflow.

Usage

```
surveyPaths(folder, projectName, date, time, run)
```

Arguments

`folder`	Character. Path to the directory where the survey is to be created.
`projectName`	Character. A string specifying the name of a project.
`date`	Numeric. Eight digit number representing a date in the format YYYYMMDD.
`time`	Numeric. Five or six digit number representing the start time of a recording session (90000 = 09:00:00, and 160000 = 16:00:00).
`run`	Numeric. Within each survey, start with run = 1, then count upwards. So, run 2 would be used to re-localize sounds that were poorly localized in run 1, etc. Running them again with slightly different settings (e.g. start/end times or low/high frequencies) can improve results.

Value

Named vector of paths to surveyFolder, runFolder, specFolder, locFolder, detectionsFile, channelsFile, and settingsFile.

Examples

```
surveyPaths(folder = tempdir(), projectName = 'Ex', date = '20200617', time = '090000', run = 1)
```

`validationSpec` *Create validation spectrograms.*

Description

This function is used inside the `localize` function to create the panels of synchronized spectrograms for manual review.

Usage

```
validationSpec(
  wavList,
  coordinates,
  locationEstimate,
  from,
  to,
  tempC = 15,
  soundSpeed,
  F_Low,
  F_High
)
```

Arguments

`wavList`	list of Wave objects. The name of the Wave objects MUST be present in the coordinates data.frame.
`coordinates`	data.frame. Must contain four required columns: column Station contains a character string with names of each recording station, while Easting, Northing and Elevation contain the x, y, and z coordinates of the station, in meters (E.g. UTM coordinates).
`locationEstimate`	Dataframe with one row containing columns Easting, Northing and Elevation, specifying the estimated location of the sound source.
`from, to`	Numeric. The portion of the wavs to plot. If missing, the whole wav will be plotted.
`tempC`	Numeric. The ambient temperature in celsius, which is used to calculate the speed of sound in air if none is specified.
`soundSpeed`	Numeric. The speed of sound. If missing, tempC will be used to calculate the speed of sound in air.
`F_Low, F_High`	Numeric. The low and high frequency, in Hz, of the sound to be localized.

Value

No return value.

Examples

```
#Get filepaths for example data.
fp <- list.files(system.file('extdata', package = 'locaR'),
                 pattern = '.mp3', full.names = TRUE)
#Add names.
names(fp) <- sapply(strsplit(basename(fp), '_'), '[[', 1)
#Load first row of detection data.
row <- read.csv(system.file('extdata',
     'Vignette_Detections_20200617_090000.csv', package = 'locaR'),
      stringsAsFactors = FALSE)[1,]
#Get non-empty Station columns.
stationSubset <- unlist(row[1,paste0('Station',1:6)])
stationSubset <- stationSubset[!is.na(stationSubset) & stationSubset != '']
#Create wav list.
wl <- createWavList(paths = fp[stationSubset], names = stationSubset,
       from = row$From, to = row$To, buffer = 0.2, index=1)
#Read coordinates.
coordinates <- read.csv(system.file('extdata', 'Vignette_Coordinates.csv',
                        package = 'locaR'), stringsAsFactors = FALSE)
row.names(coordinates) <- coordinates$Station
#Subset coordinates.
crd <- coordinates[stationSubset,]
#Localize.
loc <- localize(wavList = wl, coordinates = crd, locFolder = tempdir(),
          F_Low = row$F_Low, F_High = row$F_High, jpegName = '0001.jpeg',
          keep.SearchMap = TRUE)
#Create validation spectrogram.
#Store old par
oldpar <- par()$mfrow
par(mfrow = c(6,1))
validationSpec(wavList = wl, coordinates = crd, locationEstimate = loc$location,
                             F_Low = row$F_Low, F_High = row$F_High)
#Reset old par values.
par(mfrow = oldpar)
```

Index

checkSettings, 2
createSettings, 3, *18*, *20, 21*
createWavList, 5

getFilepaths, 6, *18*

layoutMatrix, 7
localize, 8
localizeMultiple, *4*
localizeMultiple *(*localize*)*, 8
localizeSingle, *4*, 11
locHeatmap, 12

makeSearchMap, *9*, 14
MSRP_HT_Level2 *(*MSRP_RIJ_HT*)*, 16
MSRP_Init, 15
MSRP_RIJ_HT, 16

omniSpectro, 16

parseWAFileNames, 18
processSettings, *6, 7*, *10*, *12*, *17*, 18

Rij_GCC, 19

setupSurvey, 20
spatialEntropy, 21
surveyPaths, 23

validationSpec, 24