Package: echogram (via r-universe)

October 23, 2024

Type Package

Title Echogram Visualisation and Analysis

Version 1.0.0

Date 2019-12-15

Encoding UTF-8

Author Héctor Villalobos [aut, cre]

Maintainer Héctor Villalobos <hvillalo@ipn.mx>

Description Easily import multi-frequency acoustic data stored in 'HAC' or 'RAW' files (see <http://biblio.uqar.ca/archives/30005500.pdf> for more information on the format), and produce echogram visualisations with predefined or customized color palettes. It is also possible to merge consecutive echograms; mask or delete unwanted echogram areas; model and subtract background noise; and more important, develop, test and interpret different combinations of frequencies in order to perform acoustic filtering of the echogram's data.

Depends R (>= 3.0.0)

License GPL-3

URL https://github.com/hvillalo/echogram

BugReports https://github.com/hvillalo/echogram/issues

Imports geosphere, readHAC, pals, mmand, data.table, plyr

RoxygenNote 7.2.0

Repository https://hvillalo.r-universe.dev

RemoteUrl https://github.com/hvillalo/echogram

RemoteRef HEAD

RemoteSha b211089e03abac7e287fdfc0cccbc1358ef14829

Contents

`add.echogram` *Add two echograms*

Description

This function allows addition or subtraction of Sv data matrices of corresponding echograms from two frequencies in the linear or logarithmic domain.

Usage

```
add.echogram(echogram1, echogram2, operator = c("plus", "minus"),
    domain = c("linear", "dB"))
```

Arguments

echogram1	an object of class "echogram" as returned by read.echogram.
echogram2	an object of class "echogram" from a different acoustic frequency than echogram1 above.
operator	a string indicating addition ("plus") or subtraction ("minus"). May be abbreviatted.
domain	a string indicating the domain where the operation will be performed (see details).

Details

Corresponding echograms refers to data acquired at the same time with different acoustic frequencies. In order to add echograms, the Sv data matrices must have the same dimensions. If they don't, match.echogram can be used for this purpose. It is also important to mask undesired echoes beforehand, as those belonging to the bottom and below. When domain = "dB" (the default), the Sv matrices are added as they are. When domain = "linear", the Sv values are transformed with 10^(Sv/10) before addition, and the result (X) is then back transformed to dB (10*log10(X)).

Value

An object of class "echogram" where the Sv component is the result of the performed operation.

Author(s)

Héctor Villalobos

See Also

match.echogram, mask.echogram

Examples

```
# import 38 and 120 kHz data from an HAC file
hacfile <- system.file("extdata", "D20150510-T202500.hac", package = "echogram")
echo2.038 <- read.echogram(hacfile, channel = 1)
echo2.120 <- read.echogram(hacfile, channel = 2)

## Not run:
# attempting to add the two frequencies with unequal number of pings gives an error
add.echogram(echo2.038, echo2.120, "plus", "dB")

## End(Not run)

# running match.echogram() solves this
tmp <- match.echogram(echo2.038, echo2.120)
str(tmp) # both frequencies are in a list, need to split
echo2.038 <- tmp$echogram1
echo2.120 <- tmp$echogram2

# we don't want to add bottom echoes, mask bottom and surface from both frequencies
```

```
echo2.038m <- mask.echogram(echo2.038, surf.off = 2, bott.off = 0.2)
echo2.120m <- mask.echogram(echo2.120, surf.off = 2, bott.off = 0.2)

# adding Sv values and plot result
echo.sum <- add.echogram(echo2.038m, echo2.120m, "plus", "dB")
Min <- min(as.vector(echo.sum$Sv), na.rm=TRUE) # useful to set Sv threshold
echogram(echo.sum, Svthr=floor(Min), scheme = "EK500")

# subtract 38 from 120 kHz
echo.minus <- add.echogram(echo2.120m, echo2.038m, "minus", "dB")
Min <- min(as.vector(echo.minus$Sv), na.rm=TRUE)
echogram(echo.minus, Svthr=floor(Min), scheme = "EK500")
```

`bottom.hac` *Read detected bottom range from an HAC file*

Description

This function imports, for a given acoustic channel, the detected bottom range stored in the ping tuple of an HAC file.

Usage

```
bottom.hac(hac, channel = NULL, plot = FALSE, maxDepth = NULL)
```

Arguments

`hac`	name of an HAC file.
`channel`	acoustic channel number.
`plot`	logical. if `TRUE` a plot is produced.
`maxDepth`	maximum depth (in m) represented in the plot.

Details

The acoustic channel is an integer, normally between 1 and `n`, where `n` is the number of frequencies used during data acquisition. When `channel = 1`, data from the lowest acoustic frequency is imported, while `channel = n` refers to the highest frequency present in the HAC file. By default, the function finds out the smallest channel number, because in some HAC files `channel = 0`. When a graphical representation is desired (`plot = TRUE`), the maximum displayed in the echogram depth can be provided as a negative integer with argument `maxDepth`.

Value

A data frame with two variables where every row represents one emitted ping:

`pingTime`	time of emitted ping.
`detBottom`	detected depth range in m.

Author(s)

Héctor Villalobos

Examples

```
hacfile <- system.file("extdata", "D20150510-T202221.hac", package="echogram")
bottom.hac( hacfile )
bottom.hac( hacfile, plot = TRUE )
```

---

convertAngles *Convert alongship and athwartship split beam angles*

Description

Convert angles in imported EK60 raw files from electrical to mechanical.

Usage

```
convertAngles(ekraw)
```

Arguments

ekraw	Imported EK60 data, as returned by `read.EK60_raw`.

Value

A four dimensions array [depth samples, pings, transceivers, 2] with mechanical angles.

Author(s)

Héctor Villalobos

Examples

```
if(interactive()){
ek <- read.EK60_raw("D20130504-T083828.raw", parseNMEA = TRUE, angles = TRUE)
angles <- convertAngles(ek)
dim(angles)
}
```

convertPower *Convert Received Power to Sv or TS*

Description

Convert acoustic power from imported EK60 raw files to Sv or TS data.

Usage

```
convertPower(ekraw, frequency = NULL, output = "Sv", saCorr)
```

Arguments

ekraw	Imported EK60 data, as returned by read.EK60_raw.
frequency	An integer corresponding to index of the transceiver in the data to convert.
output	A string indicating whether the power should be converted to acoustic backscattering strength ("Sv") or to target strength ("TS").
saCorr	sa correction value from calibration

Details

While the function operates on the EK60 data, it should be preferable used through ek2echogram, which will convert the ek data to echogram.

Value

A two dimensions array [depth samples, pings] with the desired output.

Author(s)

Héctor Villalobos.

See Also

ek2echogram.

Examples

```
if(interactive()){
ek <- read.EK60_raw("D20130504-T083828.raw", parseNMEA = TRUE, angles = TRUE)

Sv <- convertPower(ek, output = "Sv")
image(Sv)
}
```

dgTime *Get datagram time from imported EK* raw files*

Description

Extract datagram time and convert it to POSIXct.

Usage

```
dgTime(raw, ini)
```

Arguments

raw	A raw vector imported via read.EK_raw.
ini	Initial byte with time data.

Details

The cpu time is stored in the first 8 bytes of every datagram, after its name. It is assumed that the time zone is UTC, which may not be the case. By comparig with gps time in nmea data, a correction may be applied if needed. The index (ini) comes from output of get_dgIdx function. This function should not be called directly by the user.

Value

datagram time in POSIXct format.

Author(s)

Héctor Villalobos.

echo.noise *Sample echogram data (120 kHz)*

Description

Echogram for background noise estimation.

Usage

```
data("echo.noise")
```

Details

For this echogram, recording range (500 m) has been set above the sea bottom, and there is no evident presence of biological scatteres in order to facilitate the background noise estimation.

Examples

```
data("echo.noise")
echogram(echo.noise)
```

---

echogram *Echogram visualisation*

Description

This function allows to produce echogram visualisations from imported hac data. The user can define the visualisation Sv threshold and select between two built-in color schemes or define a custom scheme.

Usage

```
echogram(echogram, Svthr = -70, Svmax = 0, col.sep = NULL, col.nb = NULL,
         scheme = NULL, depth.grid = NULL, x.grid = NULL,
         x.ref = c("pings", "nmi", "seconds"), seabed = FALSE,
         depth.max = NULL, ping.ini = NULL, ping.max = NULL, colbar=TRUE,
         main = NULL, tformat = "%H:%M", restore.par = TRUE, ...)
```

Arguments

echogram	an object of class "echogram" as returned by read.echogram.
Svthr	Sv visualisation threshold, in decibels (dB).
Svmax	maximum Sv visualisation value, in dB.
col.sep	separation between colors. Defaults to 1 dB.
col.nb	number of colors.
scheme	color scheme for echogram, "parula" (the default), "EK500", or "echov". It can also be a vector of valid color names, or a function generating color names.
depth.grid	spacing between depth labels (in m).
x.grid	spacing between labels in the horizontal dimension according to x.ref.
x.ref	horizontal reference in echogram: "pings" (the default), "nmi" or "seconds".
seabed	logical. When TRUE and data on detected bottom is present, a line is added to the echogram.
depth.max	maximum depth to visualise.
ping.ini	initial ping to visualise.
ping.max	last ping to visualise.
colbar	logical. If TRUE a color bar is added to the echogram.
main	the acoustic frequency, by default.
tformat	time format for annotating when horizontal dimension when x.ref = 'seconds'.
restore.par	logical. If TRUE (the default), the graphical device is restored to its original state.
...	other options to image.

Details

Besides the two built-in color schemes, the user can define its own by specifying a vector of valid color names (see examples). This function uses `imageScale` function from `sinkr` package by Marc Taylor.

Author(s)

Héctor Villalobos

See Also

`palette.echogram`.

Examples

```
# import hac file
hacfile <- system.file("extdata", "D20150510-T202500.hac", package = "echogram")
echo2.038 <- read.echogram(hacfile)

# echogram by default
echogram(echo2.038)

# using alternative color schemes
echogram(echo2.038, Svthr = -70, col.sep = 1.5, scheme = "EK500")
echogram(echo2.038, Svthr = -70, col.sep = 3, scheme = c("white", "blue", "grey", "black"))
```

---

ek2echogram *Convert EK60 data to echogram*

Description

Convert imported EK60 raw data to class "echogram".

Usage

```
ek2echogram(
  ekraw,
  frequency = 1,
  data = "Sv",
  environment = NULL,
  calibration = NULL
)
```

Arguments

ekraw	Imported EK60 data, as returned by read.EK60_raw.
frequency	An integer corresponding to index of the transceiver in the data to be converted.
data	A string indicating the desired data: acoustic backscattering strength ("Sv") or target strength ("TS").
environment	An optional list with two elements: "soundVelocity" and "absorptionCoeff" calculated according to sea temperature and salinity measured during data collection.
calibration	A list as above, with "gain" and "saCorr" obtained from the echosounder calibration.

Details

This function calls the sampleRange and convertPower functions to produce an object of class "echogram".

Value

A vector with sample range (m).

Author(s)

Héctor Villalobos.

See Also

read.echogram.

Examples

```
if(interactive()){
ek <- read.EK60_raw("D20130504-T083828.raw", parseNMEA = TRUE, angles = TRUE)

eco <- ek2echogram(ek, frequency = 1, data = "Sv")
echogram(eco)
}
```

---

get_CON0 *Get CON0 datagram from EK60 raw files*

Description

Read the echosounder configuration information stored in CON0 datagram.

Usage

```
get_CON0(raw)
```

Arguments

raw	A raw vector imported via read.EK_raw or an EK60 raw file name.

Details

While it can be used for reading the configuration data from an EK60 raw file, this function is not meant to be called directly by the user.

Value

A list with two data frames: Header, containing survey-, transect- and sounder names, software version, and number of transceivers; Transceiver, with channel ID, beamtypes, frequency, gain, equivalent beam angle, etc. for each transceiver.

Author(s)

Héctor Villalobos.

Examples

```
if(interactive()){
config <- get_CON0("D20130504-T083828.raw")
config
}
```

---

get_dgIdx *Get datagram indices from EK* raw files*

Description

Find the datagram types and lengths, and their respective indices in EK* raw files.

Usage

```
get_dgIdx(raw)
```

Arguments

raw	A raw vector imported via read.EK_raw or an EK* raw file name.

Details

In EK* raw files, the first 4 bytes before every datagram contain its length (dgLen), which is repeated after the datagram data and before the length of the next datagram. So, for skipping from one datagram to the next, it takes 4 + dgLen + 4 + 1. Within each datagram, the first 4 bytes give its name (dgType: CON0, NME0, RAW0, etc.), followed by the time (next 8 bytes), and then the datagram data according to its type. The output of this fuction is key for extracting configuration data (header and transceiver information), nmea sentences, and acoustic raw data (received power and angles).

Value

A data frame with datagram types (CON0, NME0, RAW0, etc.) and lengths, and most importantly, the indices where each datagram is located.

Author(s)

Héctor Villalobos.

Examples

```
if(interactive()){
dgIdx <- get_dgIdx("D20130504-T083828.raw")
head(dgIdx)
}
```

`get_NME0` *Get NME0 datagrams from imported EK* raw files*

Description

Read the NMEA sentences stored in NME0 datagrams.

Usage

```
get_NME0(raw)
```

Arguments

`raw`	A raw vector imported via `read.EK_raw` or an EK* raw file name.

Details

The GPS data stored in the output of this function needs to be parsed to be useful. This is done with `parse.nmea()` function.

Value

A data frame with CPU time and corresponding NMEA sentences as text strings.

Author(s)

Héctor Villalobos.

See Also

`parse.nmea`

Examples

```
if(interactive()){
nmea <- get_NME0("D20130504-T083828.raw")
head(nmea)
}
```

`get_RAW0` *Get RAW0 datagrams from imported EK60 raw files*

Description

Read the acoustic sample data stored in RAW0 datagrams.

Usage

```
get_RAW0(raw, angles)
```

Arguments

`raw`	A raw vector imported via `read.EK_raw` or an EK60 raw file name.
`angles`	Logical. If `TRUE` and angle data is present, a four dimensions array containing alongship and atwarthship electrical angles is also returned.

Details

This function is not meant to be called directly by the user.

Value

A list with pings data, which includes a sample data table, and received power in a three dimensions array corresponding to the number of depth samples, number of pings, and number of transceivers. When `angles = TRUE`, alongship and athwartship electrical angles are also returned as an array with an extra dimension.

Author(s)

Héctor Villalobos.

`join.echogram` *Merge echograms*

Description

This function allows to join two echograms.

Usage

```
join.echogram(echogram1, echogram2)
```

Arguments

`echogram1`	an object of `class` "echogram" as returned by `read.echogram`.
`echogram2`	an object of `class` "echogram", preferentially contiguous in space and time with `echogram1` above.

Details

This function is designed to join echograms of the same acoustic frequency, giving an error if frequencies differ. Desirably, echograms should be contiguous in space and time, but as this is not verified, it is possible to join non-contiguous echograms.

Value

An object of `class` "echogram" resulting from the merging operation.

Author(s)

Héctor Villalobos

Examples

```
# import 38 kHz data from two consecutive HAC files
hacfile1 <- system.file("extdata", "D20150510-T202221.hac", package = "echogram")
echo1.038 <- read.echogram(hacfile1, channel = 1)

hacfile2 <- system.file("extdata", "D20150510-T202500.hac", package = "echogram")
echo2.038 <- read.echogram(hacfile2, channel = 1)

# join into one echogram
echo.038 <- join.echogram(echo1.038, echo2.038)
str(echo.038)
echogram(echo.038)
```

`mask.echogram` *Mask an echogram*

Description

This function creates, and optionally applies, a mask by "blanking" portions of the Sv data matrix of an echogram.

Usage

```
mask.echogram(echogram, surf.off = NULL, bott.off = NULL, mask = TRUE)
```

Arguments

`echogram`	an object of `class` "echogram" as returned by `read.echogram`.
`surf.off`	surface offset in m defining the upper layer (refered to the surface) to be masked.
`bott.off`	bottom offset in m defining the bottom layer (refered to the bottom) to be masked.
`mask`	logical. If `FALSE`, the function returns a masking matrix. If `TRUE` (the default), the function returns a masked echogram (see details).

Details

The masking process consists in producing a matrix of the same dimensions as the original Sv data matrix with `NA`'s in the masked portion and 1's otherwise. The product of both matrices gives the masked echogram.

Value

When `mask = FALSE`, a masking matrix is returned. When `mask = TRUE` (the default), an object of `class` "echogram" with the mask applied.

Author(s)

Héctor Villalobos

Examples

```
# import 38 kHz data from HAC file
hacfile <- system.file("extdata", "D20150510-T202500.hac", package = "echogram")
echo2.038 <- read.echogram(hacfile, channel = 1)

# make a copy of the original echogram
tmp <- echo2.038

# Create a mask, which is a matrix with 1's and NA's
mask <- mask.echogram(tmp, surf.off = 1, bott.off = -1, mask = FALSE)
image(t(mask[nrow(mask):1, ]))
```

```
# Apply mask to echogram
tmp$Sv <- tmp$Sv * mask
echogram(tmp)

# By default, the function returns the masked echogram
echo2.038mask <- mask.echogram(echo2.038, surf.off = 2, bott.off = 0.2)
echogram(echo2.038mask)
```

`match.echogram` *Match ping times from two echograms*

Description

This function verifies ping times between corresponding echograms from two frequencies, eliminating non-matching and duplicated pings.

Usage

```
match.echogram(echogram1, echogram2)
```

Arguments

`echogram1`	an object of `class` "echogram" as returned by `read.echogram`.
`echogram2`	an object of `class` "echogram" from a different acoustic frequency than `echogram1` above.

Details

Corresponding echograms refers to data acquired at the same time with different acoustic frequencies. Unmatching pings, i.e. those present in only one frequency, and duplicated pings, are identified by it's associated time and subsequently eliminated.

Value

A list with the two matched echograms.

Author(s)

Héctor Villalobos and Violeta E. González-Maynez

See Also

`add.echogram`

Examples

```
# import 38 and 120 kHz data from an HAC file
hacfile <- system.file("extdata", "D20150510-T202221.hac", package = "echogram")
echo1.038 <- read.echogram(hacfile, channel = 1)
echo1.120 <- read.echogram(hacfile, channel = 2)

# Sv matrices have different number of pings
dim(echo1.038$Sv); dim(echo1.120$Sv)

# Apply match ping times
tmp <- match.echogram(echo1.038, echo1.120)

# split the list in the two echograms
echo1.038 <- tmp$echogram1
echo1.120 <- tmp$echogram2

# number of pings and ping times are now the same for both frequencies
dim(echo1.038$Sv); dim(echo1.120$Sv)
```

`mergeSvmat` *Merge inequal Sv data matrices*

Description

Internal function called by `read.echogram` and `merge.echogram`.

Usage

```
mergeSvmat(m1, m2)
```

Arguments

`m1`	First Sv data matrix.
`m2`	Second Sv data matrix.

Author(s)

Héctor Villalobos

`navigation.hac` *Compute bearing, navigated distance and speed*

Description

This function computes navigation course (bearing), navigated distance, time diference and navigation speed between GPS fixes in position data imported from an HAC file.

Usage

```
navigation.hac(pos)
```

Arguments

`pos`	geographic position data from an HAC file, as imported with `position.hac`.

Details

The bearing and navigated distance are computed with functions `bearingRhumb` and `distVincentyEllipsoid` from package `geosphere`. This function is intended to be called inside `read.echogram`, rather than being used directly.

Value

A data frame with seven variables:

`time.cpu`	date and time from the computer CPU during data acquisition.
`lon`	longitudes.
`lat`	latitudes.
`bearing`	navigation course between two consecutive GPS fixes.
`navdist`	navigated distance between two consecutive GPS fixes.
`time.dif`	time difference between two consecutive GPS fixes.
`navspeed`	navigation speed between two consecutive GPS fixes.

Author(s)

Héctor Villalobos

See Also

`position.hac`, `bearingRhumb`, `distVincentyEllipsoid`.

Examples

```
hacfile <- system.file("extdata", "D20150510-T202221.hac", package="echogram")
pos <- position.hac( hacfile )
pos
pos2 <- navigation.hac(pos)
pos2
```

---

noise.echogram *Modelling ambient noise in echograms*

Description

This function allows to estimate a model of the background noise in an echogram by fitting the equation proposed by De Robertis and Higginbottom (2007).

Usage

```
noise.echogram(echogram, ping = NULL, dB1m = NULL, alpha = NULL, plot = TRUE, out = FALSE)
```

Arguments

echogram	an object of class "echogram".
ping	ping number for which the Sv values are to be modeled.
dB1m	noise level at 1m from the face of the transducer.
alpha	absortion coefficient of sound in sea water for echogram's frequency.
plot	logical. If TRUE (the default) a plot of the data and adjusted model is produced.
out	logical. If TRUE an echogram with the modeled noise is returned.

Details

The estimation of an ambient noise model for a particular acoustic frequency, eventually allows the "cleaning" of echograms by subtracting this noise.

Value

When plot = TRUE and out = FALSE (the default), only a plot is produced. With out = TRUE, the function returns an object of class "echogram" with the noise modelled.

Author(s)

Héctor Villalobos

References

De Robertis, A. and I. Higginbottom. 2007. A post-processing technique to estimate the signal-to-noise ratio and remove echosounder background noise 64:1282–1291.

Examples

```
# load echogram for noise estimation at 120 kHz (deep waters, no scatterers)
data("echo.noise")
attr(echo.noise$Sv, "frequency")
echogram(echo.noise, xref = "ping")

# a first look to the Sv values at ping 2
noise.echogram(echo.noise, ping = 2)

# To better adjust the model, we need to provide the absortion coefficient for 120 kHz and adjust
# the dB1m parameter. For this example, using data from a nearby CTD profile, alpha was calculated
# as being 0.03550554, while -131 dB is choosen for dB1m
noise <- noise.echogram(echo.noise, ping = 2, dB1m = -131, alpha = 0.03550554, out = TRUE)
echogram(noise)
```

---

palette.echogram *Design color palettes for echograms*

Description

This function allows to design and visualise color palettes to be used in echograms.

Usage

```
palette.echogram(Svthr, Svmax, col.sep = NULL, col.nb = NULL, scheme = NULL, visu = FALSE)
```

Arguments

Svthr	lower visualisation limit in decibels (dB).
Svmax	upper visualisation limit in dB.
col.sep	separation between colors in dB.
col.nb	number of colors.
scheme	color scheme for echogram, "parula" (the default), "EK500", or "echov". It can also be a vector of valid color names, or a function generating color names.
visu	logical. If TRUE, a visual representation of the palette is created.

Details

This function is mainly intended to be called by plot.echogram, however it is possible to use it directly in order to have a first impression of a custom color palette.

Value

A list with two elements

palette	a vector of colors
breaks	a vector of color breaks

Author(s)

Héctor Villalobos

See Also

echogram

Examples

```
palette.echogram()
palette.echogram(Svthr=-75, col.sep=1.5, scheme="EK500", visu=TRUE)
palette.echogram(Svthr=-81, col.sep=3, scheme=c("white", "blue", "black"), visu=TRUE)
```

---

parse.nmea *Parse NMEA sentences*

Description

Find and parse GPS data from NMEA sentences.

Usage

```
parse.nmea(nmea, sentence = NULL, returnAll = FALSE)
```

Arguments

nmea	A data frame with CPU time and corresponding NMEA sentences as returned by get_NME0.
sentence	A string identifying the NMEA sentence to process. It can be one of "GPRMC", "GPGGA", "INGGA", or "GPGLL".
returnAll	logical. If TRUE, all data found in the selected sentence is returned.

Details

This function looks first for the "GPRMC" sentence, if not found it tries then with "GPGGA", "INGGA", or "GPGLL", and process the one with more information (more GPS fixes). Be aware that sometimes strings may be corrupted, in which case, it is possible to choose manually the sentence to process. This is why get_NME0 does not automatically parse the NMEA strings by default. When one of "GPVTG", "INVTG", "GNVTG", or "IIVTG" sentences is found, vessel speed and bearing are also parsed and returned. Differences in time.cpu and time.gps may reveal wrong clock settings in the data aquisition computer.

Value

A data frame with cpu time, gps time (includes miliseconds if found), longitude, and latitude. Additionally, vessel speed and bearing are returned if found in the data.

Author(s)

Héctor Villalobos.

References

NMEA sentences structure can be seen in: https://gpsd.gitlab.io/gpsd/NMEA.html

See Also

get_NME0.

Examples

```
if(interactive()){
ek <- read.EK60_raw("D20130504-T083828.raw", angles = FALSE)

gps <- parse.nmea(ek$nmea)
head(gps)
}
```

---

position.hac *Read geographic position data from an HAC file*

Description

This function imports time and geographic positions recorded by a GPS in an HAC file during data acquisition.

Usage

```
position.hac(hac)
```

Arguments

hac	name of an HAC file

Details

The function looks for the Position tuple (20) in the HAC file, and if found, imports the time, latitude and longitude of GPS fixes stored in the digital echogram, as well as the CPU time of the acquisition PC.

Value

A data frame with four variables:

time.gps	date and time from the GPS during data acquisition.
time.cpu	date and time from the computer CPU during data acquisition.
lon	longitudes.
lat	latitudes.

Note

If during acoustic data acqusition the PC clock is set to UTC time, as recomended, time.gps and time.cpu will be approximately equal, because a fraction of a second is added to time.cpu to obtain a precision of 0.0001 s.

Author(s)

Héctor Villalobos

References

ICES, 2005. Description of the ICES HAC Standard Data Exchange Format, Version 1.60. Technical Report 278, ICES Cooperative Research Report.

See Also

navigation.hac

Examples

```
hacfile <- system.file("extdata", "D20150510-T202221.hac", package="echogram")
pos <- position.hac( hacfile )
pos
```

---

read.echogram *Read echogram data from an HAC file*

Description

This function imports from different tuples in the HAC file, the necessary information to visualise and analyse an echogram in R.

Usage

```
read.echogram(hac, channel = NULL)
```

Arguments

hac	name of an HAC file.
channel	acoustic channel number.

Details

This function calls internally other echogram's functions (postion.hac, navigation.hac and bottom.hac) to import data from an HAC file. The acoustic channel is an integer, normally between 1 and n, where n is the number of frequencies used during data acquisition. When channel = 1, data from the lowest acoustic frequency is imported, while channel = n refers to the highest frequency present in the HAC file. By default, the function finds out the smallest channel number, because in some HAC files channel = 0. A text string with the frequency value (in kilohertz) is stored as an attribute of the Sv matrix (see examples below).

Value

An object of class "echogram" (a list) with components:

depth	a vector of mean sample depth (in m) of length p.
Sv	a p by k matrix of sampled values, currently the mean volume backscattering strength (Sv, in dB).
pings	a k by four data frame with ping time, detected bottom depth, vessel speed and cummulated traveled distance.

Note

Currently, read.echogram has been successfully tested importing HAC data from the following ping tuples: 10000 (U-32); 10030 (U-16) and 10040 (C-16).

Author(s)

Héctor Villalobos

References

ICES, 2005. Description of the ICES HAC Standard Data Exchange Format, Version 1.60. Technical Report 278, ICES Cooperative Research Report.

Examples

```
hacfile <- system.file("extdata", "D20150510-T202221.hac", package = "echogram")
echo1 <- read.echogram(hacfile, channel = 1)
class(echo1)
str(echo1)
attr(echo1$Sv, "frequency")
echogram(echo1)
```

`read.EK60_raw` *Read raw files from Simrad EK60 scientific echosounders*

Description

This function imports all data in EK60 raw files.

Usage

```
read.EK60_raw(file, parseNMEA = FALSE, angles = TRUE)
```

Arguments

`file`	EK60 raw file name.
`parseNMEA`	logical. When false (the default) all the nmea sentences found in the raw file are returned as text strings. If `TRUE`, the `parse.nmea` function tries to parse the most appropriate nmea sentence(s) type(s).
`angles`	logical. If `TRUE` and angle data is present, a four dimensions array [depth samples, pings, transceivers, anlges] containing alongship and atwarthship electrical angles is also returned.

Details

This function imports the file as raw bytes with `read.EK_raw`. Then, the configuration information is obtained from datagram CON0 with `get_CON0`, nmea sentences (NME0) with `get_NME0`, and finally acoustic data in RAW0 datagrams with `get_RAW0`.

Value

A three elements list: (1) configuration, with Header and Transceiver(s) information; (2) nmea sentences (full or parsed); and (3) pings data, which includes at least a sample data table, and received power in a three dimensions array [depth samples, pings, transceivers]. When `angles = TRUE`, alongship and athwartship electrical angles are also returned as an array with an extra dimension.

Author(s)

Héctor Villalobos.

See Also

`read.EK_raw`, `get_CON0`, `get_NME0`, and `get_RAW0`

Examples

```
if(interactive()){
ekraw <- read.EK60_raw("D20130504-T083828.raw", parseNMEA = TRUE, angles = TRUE)

# Header
ekraw$config$Header

# Transceivers
ekraw$config$Transceiver

# GPS data
head(ekraw$nmea)
}
```

---

read.EK_bot *Read bot files from Simrad EK60 echosounder*

Description

This function will read Simrad BOT0 datagrams with bottom depth data.

Usage

```
read.EK_bot(file)
```

Arguments

file	EK60 bot file name.

Details

This function read the bot file by calling `read.EK_raw`, and `get_dgIdx` functions. From the datagram index, it finds the BOT0 datagrams, and then extracts ping times and depth(s) associated with all the transceiver(s) in the file.

Value

A data frame with ping times and detected bottom depth for every transceiver in the file.

Author(s)

Héctor Villalobos.

See Also

`read.EK_idx`.

Examples

```
fn <- system.file("extdata", "demo-D20130504-T083828.bot", package = "echogram")
depth <- read.EK_bot(fn)
head(depth)
plot(depth[, 1:2], type = "l", ylim = rev(c(0, max(depth[,2]))))
```

---

read.EK_idx *Read idx files from Simrad EK* echosounders*

Description

This function will read Simrad IDX0 datagrams with ping data.

Usage

```
read.EK_idx(file)
```

Arguments

file	EK* idx file name.

Details

This function calls `read.EK_raw`, and `get_dgIdx` functions to read the idx file, find the IDX0 datagrams, and then extracts ping information.

Value

A data frame with ping data: ping number, vessel distance, latitude, longitude, and fileoffset.

Author(s)

Héctor Villalobos.

See Also

`read.EK_bot`.

Examples

```
fn <- system.file("extdata", "demo-D20130504-T083828.idx", package = "echogram")
pings <- read.EK_idx(fn)
head(pings)
plot(pings$lon, pings$lat)
```

read.EK_raw *Read EK* raw files from Simrad echosounders*

Description

This function imports Simrad EK60 or EK80 raw files as raw bytes.

Usage

```
read.EK_raw(file)
```

Arguments

file	EK* raw file name.

Details

This function will return raw bytes as hex digits, therefore is not intended to be called directly by the user. Instead, read.EK_raw is called by other echogram's functions that also convert the various datagrams in EK60 files to their appropriate type.

Value

A vector of class "raw".

Author(s)

Héctor Villalobos.

See Also

read.EK60_raw, get_CON0, get_NME0, and get_RAW0.

sample.echogram *Select and sample data values from an echogram*

Description

This function allows to select individual pixels from an echogram and returns the Sv value, ping time and depth of the sampled pixel.

Usage

```
sample.echogram(echogram, plot = TRUE, coords = NULL, col = "black")
```

Arguments

echogram	an object of class "echogram" as returned by read.echogram.
plot	logical. If TRUE (the default), the echogram to be sampled is plotted.
coords	(x, y) coordinates (in plot units) of the desired samples. They could result from previous sampling of another frequency.
col	color for the sampled points pixels.

Details

The selection of pixels to sample can be done by clicking on the echogram or by passing the coordinates of the desired pixels to the function. The coordinates should be in plot units, and therefore, these typically come from a previous selection by clicking on another frequency's echogram (see examples).

sample.echogram makes use of locator function, and therefore it only works in devices supported by the latter, such as X11, windows and quartz.

Value

A data frame with seven variables:

id	pixel id.
x	x coordinate in plot units.
y	y coordinate in plot units.
d	distance in plot units from the selected location to a valid pixel.
pingTime	time of sampled ping.
depth	depth of the sample.
Sv	Sv value.

Author(s)

Héctor Villalobos

Examples

```
# import 38 and 120 kHz data from an HAC file
hacfile <- system.file("extdata", "D20150510-T202500.hac", package = "echogram")
echo2.038 <- read.echogram(hacfile, channel = 1)
echo2.120 <- read.echogram(hacfile, channel = 2)

# plot 38 kHz echogram
echogram(echo2.038)

## Not run:
# select points coordinates with the mouse
# click to select several locations and escape when done
pts038 <- sample.echogram(echo2.038)
pts038
```

```
# plot 120 kHz echogram
echogram(echo2.120)

# use the points previously selected for 38 kHz
pts120 <- sample.echogram(echo2.120, coords = pts038[ , 2:3])
pts120

## End(Not run)
```

`sampleRange` *Determine acoustic samples depth range*

Description

Calculate sample range according to sound speed and sample interval for imported EK60 raw files.

Usage

```
sampleRange(ekraw, frequency = NULL)
```

Arguments

`ekraw`	Imported EK60 data, as returned by `read.EK60_raw`.
`frequency`	An integer corresponding to index of the transceiver in the data for which to obtain the sample range.

Details

While the function operates on the EK60 data, it should be preferable used through `ek2echogram`, which will convert the ek data to echogram.

Value

A vector with sample range (m).

Author(s)

Héctor Villalobos.

See Also

`ek2echogram`.

Examples

```
if(interactive()){
ek <- read.EK60_raw("D20130504-T083828.raw", parseNMEA = TRUE, angles = TRUE)

R <- sampleRange(ek, frequency = 1)
R
}
```

`trim.echogram` *Trim an echogram vertically or horizontally*

Description

This function allows to trim an echogram by depth or ping number by actually trimming the underlying data matrices and vectors.

Usage

```
trim.echogram(echogram, depth.max = NULL, ping.ini = 1, ping.end = NULL)
```

Arguments

`echogram`	an object of `class` "echogram" as returned by `read.echogram`.
`depth.max`	maximum depth to keep in the echogram.
`ping.ini`	start ping to keep.
`ping.end`	end ping to keep.

Details

This funcion has been conceived to discard undesired data below a given depth (e.g. the sea bottom), therefore, the initial depth is always the surface, so the vertical trimming is limited to select the maximum depth.

Value

An object of `class` "echogram".

Author(s)

Héctor Villalobos

Examples

```
# import 38 kHz data from an HAC file
hacfile <- system.file("extdata", "D20150510-T202500.hac", package = "echogram")
echo2.038 <- read.echogram(hacfile, channel = 1)

# echogram by default
echogram(echo2.038)

# trim the echogram
echo.tmp <- trim.echogram(echo2.038, depth.max = 70, ping.end = 250)
echogram(echo.tmp)
```

`xcvrConf` *Extract transceiver configuration from EK60 raw files*

Description

Find and extract transceiver configuration information from imported EK60 raw files.

Usage

```
xcvrConf(raw, ini)
```

Arguments

`raw`	A raw vector imported via `read.EK_raw`.
`ini`	Initial byte with transceiver data

Details

The index (ini) comes from output of `get_dgIdx` function. This function should not be called directly by the user.

Value

a data frame with transceiver configuration: channel ID, beamtypes, frequency, gain, equivalent beam angle, etc. for each transceiver.

Author(s)

Héctor Villalobos

Index