Package: remora (via r-universe)

August 27, 2024

Title Rapid Extraction of Marine Observations for Roving Animals

Version 0.8-03

Date 2024-08-27

Maintainer Ian Jonsen <ian.jonsen@mq.edu.au>

Description R Package to access and integrate animal tracking data from the IMOS Animal Tracking Facility database and web-app, and environmental data from the AODN database.

URL https://github.com/IMOS-AnimalTracking/remora/, https://imos-animaltracking.github.io/remora

BugReports https://github.com/IMOS-AnimalTracking/remora/issues

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports data.table, dplyr, fasttime, furrr, future, gdistance, kableExtra, knitr, leaflet, lubridate, magrittr, ncdf4, ncmeta, parallel, doParallel, foreach, plotly, purrr, RColorBrewer, R.utils, RNetCDF, readr, sf (>= 1.0-14), shiny, stringr, tibble, tidyr, tidync, viridis, xml2, htmlwidgets, terra (>= 1.7-30), geodist (>= 0.0.8)

Suggests colormap, crosstalk, DT, geojsonio, geosphere, ggdist, gghalves, ggraph, igraph, scales, shinycssloaders, shinythemes, shinyWidgets, testthat (>= 3.0.0), tcltk, tidyverse, tools, rnaturalearth, rmarkdown, hrbrthemes, classInt, sp (>= 2.0-0)

Depends R (>= 3.6.0)

SystemRequirements GDAL (>= 3.6.2), GEOS (>= 3.11.0), PROJ (>= 9.2.0)

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/testthat/parallel true

Repository https://ocean-tracking-network.r-universe.dev

RemoteUrl https://github.com/imos-AnimalTracking/remora

RemoteRef HEAD

RemoteSha 71b90e615029bdc82454ba8b168f85f53cc7c554

Contents

`remora-package` **remora**

Description

Rapid Extraction of Marine Observations for Roving Animals

Author(s)

Ross Dwyer, Xavier Hoenner, Charlie Huveneers, Fabrice Jaine, Ian Jonsen, Francisca Maron, Kylie Scales, Vinay Udyawer

References

Hoenner, X et al. (2018) Australia's continental-scale acoustic tracking database and its automated quality control process. Scientific Data 5, 170206. https://doi.org/10.1038/sdata.2017.206

See Also

runQC extractEnv extractMoor shinyReport

`choose_file` *Choose a file interactively*

Description

Provides a platform-independent way to interactively select files

Usage

```
choose_file(caption)
```

Arguments

`caption`	caption to display in choose file window. If method defaults to base::file.choose then caption is printed to the console (doesn't work reliably due to shiny server issues)

data_with_sst_and_current *??*

Description

??

Format

.RData

det_per_receiver_location *Receiver location summary*

Description

The data needs to have the detections merged to the receiver metadata. Not to be called by user

Usage

```
det_per_receiver_location(data)
```

Arguments

data	merged detections data and receiver metadata

Value

a tibble

det_per_species *Detections per species*

Description

not to be called by user

Usage

```
det_per_species(data)
```

Arguments

data	detections data

Value

a tibble

---

det_per_station *Detections per station*

Description

not to be called by user

Usage

```
det_per_station(data)
```

Arguments

data	merged detections data and receiver metadata

Value

a tibble

---

det_per_tag *Number of detections per transmitter*

Description

not to be called by user

Usage

```
det_per_tag(data)
```

Arguments

data	detections data

Value

a tibble

det_per_tag_location *Number of detections per tag, tag deployment coordinates and date-time*

Description

not to be called by user

Usage

```
det_per_tag_location(data)
```

Arguments

data	detections data

Value

a tibble

det_species_station *Detections per station per species*

Description

not to be called by user

Usage

```
det_species_station(data)
```

Arguments

data	merged detections data and receiver metadata

Value

a tibble

`extractBlue` *Extract and append Bluelink Reanalysis (BRAN) environmental data to detection data*

Description

Accesses and download environmental data from the Bluelink (CSIRO) server and append variables to detection data based on date of detection

Usage

```
extractBlue(
  df,
  X,
  Y,
  datetime,
  env_var,
  extract_depth = 0,
  var_name = paste(env_var, extract_depth, sep = "_"),
  folder_name = "Bluelink",
  env_buffer = 1,
  cache_layers = FALSE,
  full_timeperiod = FALSE,
  station_name = NULL,
  export_step = FALSE,
  export_path = "Processed_data",
  .parallel = FALSE,
  .ncores = NULL,
  verbose = TRUE
)
```

Arguments

`df`	detection data source in data frame with at the minimum a X, Y and date time field
`X`	name of column with X coordinate or longitude (EPSG 4326)
`Y`	name of column with Y coordinate or latitude (EPSG 4326)
`datetime`	name of column with date time stamp (Coordinated Universal Time; UTC)
`env_var`	variable needed from Bluelink. Options include ('BRAN_temp', 'BRAN_salt', BRAN_ssh', 'BRAN_mld', 'BRAN_cur', 'BRAN_wind').
`extract_depth`	Bluelink data is 3D, so data can be obtained either at the water surface or at depth. Please provide the depth of interest (between 0 and 4,509 m) as numeric and the function will automatically obtain the data at the nearest available layer. By default the data will be extracted at the water surface.

`var_name` name for the column including the extracted environmental data. Can be usefull if the user wants to download the same environmental data at different depths. If not specified, it will be chosen based on the env_var and extract_depth arguments.

`folder_name` name of folder within the working directory where the downloaded and processed netCDF files should be saved. Default (NULL) produces automatic folder names based on study extent and deletes processed files after processing.

`env_buffer` distance (in decimal degrees) to expand the study area beyond the coordinates to extract environmental data. Default value is 1°.

`cache_layers` should the downloaded and processed environmental data be cached within the working directory? If FALSE (default), the Bluelink data will be stored in a temporary folder and discarded after environmental extraction. Otherwise, it will be saved in the "cached" folder within folder_name.

`full_timeperiod` should environmental variables be extracted for each day across full monitoring period? This option is time and memory consuming for long projects. If this option is selected, the returned dataset will be standardized for the days with/without detections across all stations (station_name column) where animals were detected. For more details please see the package vignettes.

`station_name` if full_timeperiod = TRUE, please provide the column that identifies the name of the acoustic stations

`export_step` should the processed dataset be exported to file? This is particularly usefull for large datasets, to avoid function failure due to issues with internet connexion. The rows with missing data will be exported as NAs, and only these will be completed if the function is rerun providing the exported dataset as input (df)

`export_path` path and name of to export the dataset with appended environmnetal data

`.parallel` should the function be run in parallel

`.ncores` number of cores to use if set to parallel. If none provided, uses `detectCores` to determine number.

`verbose` should function provide details of what operation is being conducted. Set to `FALSE` to keep it quiet

Details

The `extractBlue` function allows the user to download, process and append a range of 3D environmental variables (between the water surface to 4,509 m depth) to each detection within a telemetry data set. We advocate for users to first undertake a quality control step using the `runQC` function before further analysis, however the functionality to append environmental data will work on any dataset that has at the minimum spatial coordinates (i.e., latitude, longitude; in EPSG 4326) and a timestamp (in UTC) for each detection event. Quality controlled environmental variables housed in the Bluelink (BRAN) CSIRO server will be extracted for each specific coordinate at the specific timestamp where available. A summary table of the full range of environmental variables currently available can be accessed using the `imos_variables` function.

Value

a dataframe with the environmental variable appended as an extra column based on date of each detection

Examples

```
## Input example detection dataset that have run through the quality control
##   workflow (see 'runQC' function)

library(tidyverse)
data("TownsvilleReefQC")

## simplify & subset data for speed
qc_data <-
  TownsvilleReefQC %>%
  unnest(cols = c(QC)) %>%
  ungroup() %>%
  filter(Detection_QC %in% c(1,2)) %>%
  filter(filename == unique(filename)[1]) %>%
  slice(1:20)

## Extract daily interpolated sea surface temperature
## cache_layers & fill_gaps args set to FALSE for speed
data_with_temp <-
   extractBlue(df = qc_data,
               X = "receiver_deployment_longitude",
               Y = "receiver_deployment_latitude",
               datetime = "detection_datetime",
               env_var = "BRAN_temp",
               extract_depth = 0,
               verbose = TRUE)
```

`extractEnv` *Extract and append remote sensing environmental data to detection data*

Description

Accesses and download environmental data from the IMOS THREDDS server and append variables to detection data based on date of detection

Usage

```
extractEnv(
  df,
  X = "longitude",
  Y = "latitude",
  datetime = "detection_timestamp",
```

```
  env_var,
  folder_name = NULL,
  verbose = TRUE,
  cache_layers = TRUE,
  crop_layers = TRUE,
  full_timeperiod = FALSE,
  fill_gaps = FALSE,
  buffer = NULL,
  nrt = FALSE,
  output_format = ".grd",
  .parallel = TRUE,
  .ncores = NULL
)
```

Arguments

df	detection data source in data frame with at the minimum a X, Y and date time field
X	name of column with X coordinate or longitude (EPSG 4326)
Y	name of column with Y coordinate or latitude (EPSG 4326)
datetime	name of column with date time stamp (Coordinated Universal Time; UTC)
env_var	variable needed options include ('rs_sst', 'rs_sst_interpolated', 'rs_salinity', 'rs_chl', 'rs_turbidity', 'rs_npp', 'bathy', 'dist_to_land', 'rs_current')
folder_name	name of folder within 'imos.cache' where downloaded rasters should be saved. default NULL produces automatic folder names based on study extent
verbose	should function provide details of what operation is being conducted. Set to FALSE to keep it quiet
cache_layers	should the extracted environmental data be cached within the working directory? if FALSE stored in temporary folder and discarded after environmental extraction
crop_layers	should the extracted environmental data be cropped to within the study site
full_timeperiod	should environmental variables extracted for each day across full monitoring period, time and memory consuming for long projects
fill_gaps	should the function use a spatial buffer to estimate environmental variables for detections where there is missing data. Default is FALSE to save computational time.
buffer	radius of buffer (in m) around each detection from which environmental variables should be extracted from. A median value of pixels that fall within the buffer will be used if fill_gaps = TRUE. If NULL a buffer will be chosen based on the resolution of environmental layer. A numeric value (in m) can be used here to customise buffer radius.
nrt	should Near Real-Time current data be used if Delayed-Mode current data is missing. Default is FALSE, in which case NA's are appended to current variables for years (currently, all years after 2020) when current data are missing. Note

	that Near Real-Time data are subject to less quality control than Delayed-Mode data.
output_format	File format for cached environmental layers. You can use gdal(drivers=TRUE) to see what drivers are available in your installation. The default format is '.grd'.
.parallel	should the function be run in parallel
.ncores	number of cores to use if set to parallel. If none provided, uses detectCores to determine number.

Details

The extractEnv function allows the user to access, download and append a range of environmental variables to each detection within a telemetry data set. We advocate for users to first undertake a quality control step using the runQC function before further analysis, however the functionality to append environmental data will work on any dataset that has at the minimum spatial coordinates (i.e., latitude, longitude; in EPSG 4326) and a timestamp (in UTC) for each detection event. Quality controlled environmental variables housed in the IMOS Thredds server will be extracted for each specific coordinate at the specific timestamp where available. A summary table of the full range of environmental variables currently available can be accessed using the imos_variables function.

Value

a dataframe with the environmental variable appended as an extra column based on date of each detection

Examples

```
## Input example detection dataset that have run through the quality control
##   workflow (see 'runQC' function)

library(tidyverse)
data("TownsvilleReefQC")

## simplify & subset data for example speed-up
qc_data <-
  TownsvilleReefQC %>%
  unnest(cols = c(QC)) %>%
  ungroup() %>%
  filter(Detection_QC %in% c(1,2)) %>%
  filter(filename == unique(filename)[1]) %>%
  slice(5:8)

## Extract daily interpolated sea surface temperature
## cache_layers & fill_gaps args set to FALSE for speed
data_with_sst <-
  extractEnv(df = qc_data,
             X = "receiver_deployment_longitude",
             Y = "receiver_deployment_latitude",
             datetime = "detection_datetime",
             env_var = "rs_sst_interpolated",
             cache_layers = FALSE,
```

```
crop_layers = TRUE,
full_timeperiod = FALSE,
fill_gaps = TRUE,
folder_name = "test",
.parallel = FALSE)
```

`extractMoor` *Link the QC Detection Dataset to the Sensor Value obtained by the Nearest IMOS Mooring*

Description

Extracts specified sensor value(s) observed by the IMOS mooring nearest to the supplied QC'd detection location(s)

Usage

```
extractMoor(
  trackingData,
  file_loc,
  sensorType = "temperature",
  timeMaxh = Inf,
  distMaxkm = Inf,
  targetDepthm = NA,
  scalc = c("min", "max", "mean")
)
```

Arguments

`trackingData`	dataframe containing acoustic detection data in IMOS QC format with the `moor_site_code` next to it as generated using the `mooringDistance` function.
`file_loc`	character string describing folder location in which NetCDF files of mooring data have been saved after running mooringDownload function
`sensorType`	character string containing name of mooring sensor to query. Can be "temperature", "velocity" or "salinity".
`timeMaxh`	optional numeric string containing the maximum time threshold to merge detection and mooring sensor values.
`distMaxkm`	optional numeric string containing the maximum distance threshold in kilometers to include in output.
`targetDepthm`	extracts the nearest sensor to this depth value. if set as NA then all sensors returned for this timestamp at this mooring
`scalc`	select the lower or higher depth value when there are 2 options for sensors nearest the targetDepth provided. Users can also specify a mean or `NA` to return all sensor values.

Value

the trackingData dataframe as a nested tibble object with the sensor values from the nearest moorLocations that fall within the specified time, distance and depth thresholds.

---

getDistance *Find closest mooring*

Description

Links the QC Detection Dataset to the Nearest IMOS Mooring ID

Usage

```
getDistance(
  trackingData,
  moorLocations,
  X = "receiver_deployment_longitude",
  Y = "receiver_deployment_latitude",
  datetime = "detection_datetime"
)
```

Arguments

trackingData	dataframe containing acoustic detection data in IMOS QC format
moorLocations	dataframe containing the locations of IMOS moorings
X	name of column with X coordinate or longitude (EPSG 4326)
Y	name of column with Y coordinate or latitude (EPSG 4326)
datetime	name of column with date time stamp (Coordinated Universal Time; UTC)

Value

The trackingData dataframe with the nearest moor_site_code to each acoustic detection

---

getOverlap *Extract number of tag detections that fall within the duration of sensor coverage at each mooring*

Description

Extract number of tag detections that fall within the duration of sensor coverage at each mooring

Usage

```
getOverlap(x)
```

Arguments

x	The trackingData dataframe with the nearest moor_site_code to each acoustic detection

Value

a grouped tibble describing proportion of temporal overlap between mooring coverage and detections dataset

---

get_expert_distribution_shp_CAAB

match species CAAB code with CSIRO expert distribution shapefiles

---

Description

Get expert distribution shapefile, if available, from CSIRO's Geoserver. Full CAAB code list: http://www.marine.csiro.au/datacentre/caab/caab_dump_latest.xlsx

Usage

```
get_expert_distribution_shp_CAAB(CAAB_species_id, spe)
```

Arguments

CAAB_species_id	CAAB id of species for which a distribution shapefile is required
spe	species scientific name

Details

For a few species acoustically tagged no shapefile exists.

Value

shp is a multipolygon sf data.frame object of the species' distribution

Examples

```
# example code
x <- TownsvilleReefQC$QC[[1]]
expert_shp <- get_expert_distribution_shp_CAAB(CAAB_species_id = x$CAAB_species_id[1],
spe = x$species_scientific_name[1])
```

grabQC *grab subsets of the QC output*

Description

grabQC() lets you obtain subsets of the QC output

Usage

```
grabQC(
  x,
  what = c("dQC", "detections", "QCflags", "tag_meta", "rec_meta", "meas"),
  flag = "all"
)
```

Arguments

x	a QC output object (a nested tibble with class remora_QC)
what	defined subset of the QC output is to be grabbed; either detections, QCflags, dQC (detections and QCflags), tag_meta (transmitter deployment metadata), rec_meta (receiver deployment metadata), or meas (animal measurements)
flag	specifies which quality controlled detections to return (see examples): any combination of: valid, likely valid, likely invalid, invalid, or all. The default is to return all detections. Ignored if what is any of tag_meta, rec_meta or meas.

Value

a data frame with the requested subset of the QC output

Examples

```
## grab detections and QCflags from example QC output & return only the
## `valid` and `likely valid` detections
data(TownsvilleReefQC)
d.qc <- grabQC(TownsvilleReefQC, what = "dQC", flag = c("valid", "likely valid"))

## return all detections
d.qc <- grabQC(TownsvilleReefQC, what = "dQC")
```

imos_variables *Table of all available IMOS variables accessible from* remora *package*

Description

Summary of available IMOS environmental variable to append to detection data.

Usage

```
imos_variables(variable = NULL)
```

Arguments

variable	(optional) name of a specific variable you are interested in. Default of NULL provides details for all available datasets.

Details

This function helps users identify what quality controlled environmental variables can be accessed and appended to their detection data using the extractEnv of mooringTable functions.

Value

a formatted table with details of available variables accessible through the remora package

Examples

```
## Identify all available variables
imos_variables()

## If there is a specific variable you are interested in
imos_variables(variable = "rs_sst_interpolated")
```

imos_variables_table *??*

Description

??

Format

.RData

mooringDownload *Download the IMOS Mooring Data for a Single Mooring Site*

Description

Downloads all IMOS mooring data for a specified, single mooring site

Usage

```
mooringDownload(
  moor_site_codes,
  fromWeb,
  sensorType = "temperature",
  itimeout = 240,
  file_loc
)
```

Arguments

moor_site_codes	character vector of the name(s) of the desired mooring(s) (updated from moorID to match download file)
fromWeb	logical string detailing whether or no the netCDF should be downloaded from the web or uploaded from a saved file location
sensorType	character string detailing which sensor value we are interested in. Can be "temperature", "velocity", "salinity or "oxygen"."
itimeout	integer value for number of seconds we are willing to wait before timeout to download netcdf from the web. Defaults to 60
file_loc	character string for the location of the saved files

Value

The moorData dataframe with the sensor information time series for a given site_code

mooringTable *Pull the Latest IMOS Moorings Dataset from the Server*

Description

Uses a Web link to send a Web Feature Service (WFS) query directly to AODN geoserver

Usage

```
mooringTable(sensorType = "temperature")
```

Arguments

sensorType	Character string containing name of mooring sensor to query. Can be "temperature" or "velocity"

Value

dataframe containing the locations of IMOS moorings as csv tibble

---

n_detections *Number of total detections per transmitter*

Description

not to be called by user

Usage

```
n_detections(data)
```

Arguments

data	detections data

---

n_receivers *Number of total receivers*

Description

not to be called by user

Usage

```
n_receivers(data)
```

Arguments

data	merged detections data and receiver metadata

n_species *Number of total species*

Description

not to be called by user

Usage

```
n_species(data)
```

Arguments

data	detections data

n_tags *Number of total transmitters*

Description

not to be called by user

Usage

```
n_tags(data)
```

Arguments

data	detections data

plotDT *Depth-time plot*

Description

Creates an interactive depth-time plot from a sensor dataset from a specified IMOS mooring, with an option to overlay species detection records from tracking dataset from closest receiver

Usage

```
plotDT(
  moorData,
  moorName,
  dateStart = NULL,
  dateEnd = NULL,
  varName = c("temperature", "vcur", "ucur", "psal"),
  trackingData = NULL,
  speciesID,
  IDtype = "CAAB",
  detStart = NULL,
  detEnd = NULL
)
```

Arguments

moorData	Dataframe containing the sensor data from a single IMOS mooring.
moorName	Character string specifying mooring name using IMOS ID code.
dateStart	Optional character string of start date of date range of mooring data to plot, as "YYYY-MM-DD"
dateEnd	Optional character string of end date of date range of mooring data to plot, as "YYYY-MM-DD"
varName	Character string of mooring sensor variable of interest. Can be "temperature", "ucur" for meridional (u) velocity, "vcur" for zonal (v) velocity
trackingData	Dataframe containing tag detections data to plot. Takes output from getDistance() function as input. Set as NULL for a base plot with no detections.
speciesID	Character string describing species or tag identifier.
IDtype	Character string describing type of species or tag identifier. Can be CAAB code, tag ID, common name or scientific name. Must match identifier in detections dataset.
detStart	Optional character string of start date of date range of detections data to overlay on depth-time plot.
detEnd	Optional character string of end date of date range of detections data to overlay on depth-time plot.

---

plotQC *produce interactive leaflet maps showing the occurrence of QC'd detections per species*

Description

`plotQCint()` QC'd detections colour-coded by their assessed validity status, overlaid on species expert distribution extent

Usage

```
plotQC(x, path = NULL, pal = "PuOr", revpal = TRUE)
```

Arguments

x	a remora output object with `class(remora_QC)`.
path	path to save map(s). Options are: `NULL` (default) - renders to a viewer window (RStudio only); `wb` - renders in default web browser; or a valid file path - map saved as a self-contained .html file.
pal	a `brewer.pal` palette name as a quoted character string. Use `RColorBrewer::display.brewer.all()` to see choices.
revpal	reverse order of colour palette.

Value

produces interactive leaflet maps of species expert distribution and location of QC'd detections

Examples

```
## example QC'd data
data(TownsvilleReefQC)

## save plot as an .html file to the working directory
plotQC(TownsvilleReefQC, path = ".")

## clean up
system("rm *_QCmap.html")
```

---

remoteNCDF *Remotely open BRAN netCDF files*

Description

Open netCDF files from BRAN and converts them into data frame format.

Usage

```
remoteNCDF(year, month, var, depth, lon.min, lon.max, lat.min, lat.max)
```

Arguments

year	Year of interest for data download
month	Month of interest for data download
var	Variable of interest for data download
depth	Depth of interest for data download
lon.min	Minimum longitude for data download
lon.max	Maximum longitude for data download
lat.min	Minimum latitude for data download
lat.max	Maximum latitude for data download

Details

Returns a dataframe with environmental data

---

runQC *run IMOS-ATF acoustic detections quality control process*

Description

conduct quality control (QC) on IMOS-ATF acoustic detections data

Usage

```
runQC(
  x,
  lat.check = TRUE,
  .parallel = FALSE,
  .ncores = detectCores() - 2,
  .progress = TRUE
)
```

Arguments

x	a 4-element list of paths to detections, receiver and transmitter deployment metadata, and animal measurements data files. These data must be downloaded from the IMOS-ATF Web App (URL), or have exactly the same structure and variable names as the Web App data.
lat.check	(logical; default TRUE) test for receiver_deployment_latitudes in N hemisphere at correct to S hemisphere. Set to FALSE if QC'ing N hemisphere data
.parallel	logical; run QC tests in parallel across multiple processors (default is FALSE)
.ncores	integer; number of cores to run in parallel. If NULL and `parallel = TRUE` then process will run across all available cores, otherwise run across user-specified cores
.progress	logical; display QC progress (default is TRUE).

Details

The QC process merges data from the supplied files downloaded via the IMOS-ATF Web App (URL): `IMOS_detections.csv`; `IMOS_receiver_deployment_metadata.csv`; `IMOS_transmitter_deployment_metadat` and `IMOS_animal_measurements.csv`. Eight quality control tests are performed on the detections, as outlined in Hoenner et al. (2018), and QC flags are appended to the merged data for each of these 8 tests.

The QC flags are values ranging between 1 and 4, representing `valid`, `likely valid`, `likely invalid`, and `invalid` detections, respectively. The user can then employ these flags, in various combinations, to filter the merged data (see `examples` & `vignette`).

Utility functions are provided to subset the merged data in various ways from the nested tibble and to visualise the QC results (see `examples` & `vignette`).

A QC log is written to `QC_logfile.txt` in the working directory. The logfile documents potential data issues discovered during the QC process: e.g., `transmitter_deployment_id`'s present in the detections file but not in the transmitter metadata file (if supplied); `receiver_deployment_id`'s present in the detections file but not in the receiver metadata file (if supplied); NA's present in transmitter deployment locations. Generally, these issues can not be corrected automatically and require the user to investigate the cause and take appropriate steps to correct the data.

Value

the QC output is returned to the parent frame as a nested tibble with class `remora_QC`

References

Hoenner, X et al (2018) Australia's continental-scale acoustic tracking database and its automated quality control process. Scientific Data 5, 170206. https://doi.org/10.1038/sdata.2017.206

Examples

```
## specify files to QC - use supplied example .csv data
files <- list(det = system.file(file.path("test_data","IMOS_detections.csv"),
                      package = "remora"),
              rmeta = system.file(file.path("test_data",
              "IMOS_receiver_deployment_metadata.csv"),
                      package = "remora"),
              tmeta = system.file(file.path("test_data",
              "IMOS_transmitter_deployment_metadata.csv"),
                      package = "remora"),
              meas = system.file(file.path("test_data",
              "IMOS_animal_measurements.csv"),
                      package = "remora"))
qc.out <- runQC(files)
plotQC(qc.out, path = ".") # saves .html file to working directory

## get detections with QC flags
d.qc <- grabQC(qc.out, what = "dQC")

## clean up
system("rm QC_logfile.txt *_QCmap.html")
```

sei *Station Efficiency Index*

Description

not to be called by user

Usage

```
sei(data, date1, date2)
```

Arguments

data	merged detections data and receiver metadata
date1	start date for SEI calculation
date2	end date for SEI calculation

Value

a tibble

shinyReport *Render Shiny report*

Description

Renders the Shiny App for transmitter and receiver reports.

Usage

```
shinyReport(type = "transmitters")
```

Arguments

type	"transmitters" or "receivers" to produce the corresponding report

Value

Shiny app with Transmitter or Receiver Project visualisations and statistics

Examples

```
## Not run:
shinyReport(type = "transmitters")

## End(Not run)
```

TownsvilleReefQC *Quality-controlled bull shark detections (5 individuals, subsampled for efficiency)*

Description

Example bull shark acoustic tracking data. Data were sourced from the Integrated Marine Observing System (IMOS) - IMOS is supported by the Australian Government through the National Collaborative Research Infrastructure Strategy and the Super Science Initiative.

Format

.RData

windDir *Calculate wind direction*

Description

helper function to calculate wind direction

Usage

```
windDir(u, v)
```

Arguments

u	horizontal (u) wind speed
v	vertical (v) wind speed

Value

Wind direction in degrees clockwise

windSpd *Calculate wind speed*

Description

helper function to calculate wind speed

Usage

```
windSpd(u, v)
```

Arguments

u	horizontal (u) wind speed
v	vertical (v) wind speed

Value

Wind speed in meters per second

writeQC *write a QCsummary.csv file and/or QC processed* .csv *files for each tag deployment*

Description

writes a single QCsummary.csv file and/or tag-deployment-specific `temporal_outcome`'s of quality-controlled detections to `.csv` files

Usage

```
writeQC(x, path = NULL, summary = TRUE, csv = TRUE)
```

Arguments

x	a nested tibble with class `remora_QC` generated by `runQC`
path	path to write QC'd files to. If NULL the files are written to the working directory
summary	whether to generate a summary `.csv` file. If TRUE then `QCsummary.csv` is written to the specified `path`
csv	whether to generate the individual `.csv` files. If TRUE (default) then each file is written to the specified `path`

Details

takes a `remora_QC` nested tibble and writes .csv files corresponding to each row of the nested tibble. A summary of the QC process is written to `QCsummary.csv`. The summary contains the following information for each QC'd tag (see references for more details):

- total_detectionsthe number of raw detections
- detections_before_deploymentthe number of detections recorded prior to deployment date in metadata
- invalid_deployment_locationlogical indicating whether deployment longitude,latitude is valid
- detections_outside_species_rangethe number of detections that occurred outside the species expert distribution
- valid_detectionsthe number of detections that passed the QC process (QC flags 1 and 2)
- tracking_duration_daysthe number of days between deployment and the last valid detection
- invalid_velocitythe number of detections associated with implausible travel speeds

Value

.csv files and/or a QCsummary.csv file are written to the specified `path`

References

Hoenner X et al. (2018) Australia's continental-scale acoustic tracking database and its automated quality control process. Sci Data 5, 170206 https://doi.org/10.1038/sdata.2017.206

Examples

```
## Not run:
## example QC'd data
data(TownsvilleReefQC)
## write QC output
writeQC(TownsvilleReefQC, summary = TRUE)

## End(Not run)
```

Index