Package 'swfscDAS'

September 3, 2021

Title Southwest Fisheries Science Center Shipboard DAS Data Processing

Version 0.6.0

Description Process and summarize shipboard
'DAS' <https://swfsc-publications.fisheries.noaa.gov/publications/TM/SWFSC/NOAA-TM-NMFS-SWFSC-305.PDF> data
produced by the Southwest Fisheries Science Center (SWFSC) program 'WinCruz'
<https://www.fisheries.noaa.gov/west-coast/science-data/california-current-marine-mammal-assessment-program>.
This package standardizes and streamlines basic DAS data processing,
and includes a PDF with the DAS data format requirements.

URL https://smwoodman.github.io/swfscDAS/,
https://github.com/smwoodman/swfscDAS/

BugReports https://github.com/smwoodman/swfscDAS/issues/

Depends R (>= 4.0.0)

Imports dplyr, lubridate, magrittr, methods, parallel, purrr, readr,
rlang, sf, swfscMisc, tidyr

Suggests knitr, rmarkdown, stringr, testthat (>= 2.1.0)

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Sam Woodman [aut, cre] (<https://orcid.org/0000-0001-6071-8186>)

Maintainer Sam Woodman <sam.woodman@noaa.gov>

Repository CRAN

Date/Publication 2021-09-03 19:40:02 UTC

R topics documented:

`swfscDAS-package` *Southwest Fisheries Science Center DAS*

Description

Process and summarize shipboard DAS data

Details

This package contains functions designed for processing and analyzing DAS data generated using the WinCruz program by the Southwest Fisheries Science Center. It is intended to standardize and streamline basic DAS data processing.

Author(s)

Sam Woodman <sam.woodman@noaa.gov>

See Also

https://smwoodman.github.io/swfscDAS/

as_das_df *Coerce object to a das_df object*

Description

Check if an object is of class das_df, or coerce it if possible.

Usage

```
as_das_df(x)

## S3 method for class 'das_df'
as_das_df(x)

## S3 method for class 'data.frame'
as_das_df(x)
```

Arguments

x	an object to be coerced to class das_df

Details

Only data frames can be coerced to an object of class das_df. If x does not have column names and classes as specified in das_df-class, then the function returns an error message detailing the first column that does not meet the requirements of a das_df object.

Value

An object of class ‘das_df‘

See Also

das_df-class

as_das_dfr *Coerce object to a das_dfr object*

Description

Check if an object is of class das_dfr, or coerce it if possible.

Usage

```
as_das_dfr(x)

## S3 method for class 'das_dfr'
as_das_dfr(x)

## S3 method for class 'data.frame'
as_das_dfr(x)
```

Arguments

x	an object to be coerced to class das_dfr

Details

Only data frames can be coerced to an object of class das_dfr. If x does not have column names and classes as specified in das_dfr-class, then the function returns an error message detailing the first column that does not meet the requirements of a das_dfr object.

Value

An object of class 'das_dfr'

See Also

das_dfr-class

---

das_check *Check DAS file*

Description

Check that DAS file has accepted formatting and values

Usage

```
das_check(
  file,
  skip = 0,
  file.out = NULL,
  sp.codes = NULL,
  print.cruise.nums = TRUE
)
```

Arguments

`file`	filename(s) of one or more DAS files
`skip`	integer: see `read_fwf`. Default is 0
`file.out`	filename to which to write the error log; default is `NULL`
`sp.codes`	character; filename of .dat file from which to read accepted species codes. If `NULL`, species codes will not be checked. Default is `NULL`
`print.cruise.nums`	logical; indicates if a table with all the cruise numbers in the `x` should be printed using `table`. Default is `TRUE`

Details

Precursor to a more comprehensive DASCHECK program. This function checks that the following is true:

- Event codes are one of the following: #, *, ?, 1, 2, 3, 4, 5, 6, 7, 8, A, B, C, E, F, k, K, N, P, Q, r, R, s, S, t, V, W, g, G, p, X, Y, Z
- Latitude values are between -90 and 90 (inclusive; NA values are ignored)
- Longitude values are between -180 and 180 (inclusive; NA values are ignored)
- The effort dot matches effort determined using B, R, and E events
- There are an equal number of R and E events, and they alternate occurrences
- A BR event series or R event does not occur while already on effort
- An E event does not occur while already off effort
- All Data# columns for non-C events are right-justified
- Only C events have data past the 99th column in the DAS file
- The following events have NA (blank) Data# columns: *
- All of *, B, R, E, V, W, N, P, and Q events have NA Data# columns where specified (see format pdf for more details)
- Event/column pairs meet the following requirements:

Item	*Event*	*Column*	*Requirement*
Cruise number	B	Data1	Can be converted to a numeric value
Mode	B	Data2	Must be one of C, P, c, p, or NA (blank)
Echo sounder	B	Data4	Must be one of Y, N, y, n, or NA (blank)
Effort type	R	Data1	Must be one of F, N, S, or NA (blank)
ESW sides	R	Data2	Effective strip width; must be one of F, H, or NA (blank)
Course	N	Data1	Can be converted to a numeric value
Speed	N	Data2	Can be converted to a numeric value
Beaufort	V	Data1	Must be a whole number between 0 and 9
Swell height	V	Data2	Can be converted to a numeric value
Wind speed	V	Data5	Can be converted to a numeric value
Rain or fog	W	Data1	Must be between 0 and 5 and either a whole number or have decima
Horizontal sun	W	Data2	Must be a whole number between 0 and 12
Vertical sun	W	Data3	Must be a whole number between 0 and 12

Visibility	W	Data5	Can be converted to a numeric value
Sighting (mammal)	S, K, M	Data3-7	Can be converted to a numeric value
Sighting (mammal)	G	Data5-7	Can be converted to a numeric value
Sighting cue (mammal)	S, K, M	Data3	Must be a whole number between 1 and 6
Sighting method (mammal)	S, K, M, G	Data4	Must be a whole number between 1 and 7
Bearing (mammal)	S, K, M, G	Data5	Must be a whole number between 0 and 360
Photos	A	Data3	Must be one of N, Y, n, y, or NA (blank)
Birds	A	Data4	Must be one of N, Y, n, y, or NA (blank)
Calibration school	S, K, M	Data10	Must be one of N, Y, n, y, or NA (blank)
Aerial photos taken	S, K, M	Data11	Must be one of N, Y, n, y, or NA (blank)
Biopsy taken	S, K, M	Data12	Must be one of N, Y, n, y, or NA (blank)
Species codes	A	Data5-8	If a species codes file is provided, must be one of the provided codes
Resight	s, k	Data2-5	Can be converted to a numeric value
Turtle species	t	Data2	If a species codes file is provided, must be one of the provided codes
Turtle sighting	t	Data3-5, 7	Can be converted to a numeric value
Turtle JFR	t	Data6	Must be one of F, J, N, R, or NA (blank)
Fishing vessel	F	Data2-4	Can be converted to a numeric value
Sighting info	1-8	Data2-8	Can be converted to a numeric value
Sighting info	1-8	Data9	The Data9 column must be NA (blank) for events 1-8

In the table above, 'between' means inclusive.

Long-term items, and checks that are not performed:

- Check that datetimes are sequential, meaning they 1) are the same as or 2) come after the previous event
- Check that A events only come immediately after a G/S/K/M event, and all G/S/K/M events have an A after them. And that each has at least one group size estimate (1:8 event)

Value

A data frame with columns: the file name, line number, cruise number, 'ID' (columns 4-39 from the DAS file), and description of the issue

If `file.out` is not `NULL`, then the error log data frame is also written to `file.out` using `write.csv`

A warning is printed if any events are r events; see `das_process` for details about r events

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
if (interactive()) das_check(y)
```

das_chop_condition *Chop DAS data - condition*

Description

Chop DAS data into a new effort segment every time a specified condition changes

Usage

```
das_chop_condition(x, ...)

## S3 method for class 'data.frame'
das_chop_condition(x, ...)

## S3 method for class 'das_df'
das_chop_condition(
  x,
  conditions,
  seg.min.km = 0.1,
  distance.method = NULL,
  num.cores = NULL,
  ...
)
```

Arguments

x	an object of class das_df, or a data frame that can be coerced to class das_df. This data must be filtered for continuous effort sections; see the Details section below
...	ignored
conditions	the conditions that trigger a new segment; see das_effort
seg.min.km	numeric; minimum allowable segment length (in kilometers). Default is 0.1. See the Details section below for more information
distance.method	character; see das_effort. Default is NULL since these distances should have already been calculated
num.cores	see das_effort

Details

WARNING - do not call this function directly! It is exported for documentation purposes, but is intended for internal package use only.

This function is intended to be called by das_effort when the "condition" method is specified. Thus, x must be filtered for events (rows) where either the 'OnEffort' column is TRUE or the 'Event' column is "E"; see das_effort for more details. This function chops each continuous effort section

(henceforth 'effort sections') in `x` into modeling segments (henceforth 'segments') by creating a new segment every time a specified condition changes. Each effort section runs from an "R" event to its corresponding "E" event. After chopping, `das_segdata` is called (with `segdata.method = "maxdist"`) to get relevant segdata information for each segment.

Changes in the one of the conditions specified in the `conditions` argument triggers a new segment. One exception is if the event at which this condition change occurs is part of an event series, meaning one of several events in a row at the same lat/lon points (such as a PVNW event series). In this situation, the final event of the event series is considered the last event of the current effort segment, and thus also the start of the next effort segment.

Related, when multiple condition changes happen at the same lat/lon points, such as a "RPVNW" series of events at the beginning of the effort section. When this happens, no segments of length zero are created; rather, a single segment is created that includes all of the condition changes (i.e. all of the events in the event series) that happened during the series of events (i.e. at the same location). Note that this combining of events at the same position happens even if `seg.min.km = 0`.

In addition, (almost) all segments whose length is less than `seg.min.km` are combined with the segment immediately following them to ensure that the length of (almost) all segments is at least `seg.min.km`. This allows users to account for situations where multiple conditions, such as Beaufort and the visibility, change in rapid succession, for instance <0.1 km apart. When segments are combined, a message is printed, and the condition that was recorded for the maximum distance within the new segment is reported. See `das_segdata`, `segdata.method = "maxdist"`, for more details about how the segdata information is determined. The only exception to this rule is if the short segment ends in an "E" event, meaning it is the last segment of the effort section. Since in this case there is no 'next' segment, this short segment is left as-is.

If the column `dist_from_prev` does not exist, the distance between subsequent events is calculated as described in `das_effort`

Value

List of two data frames:

- `x`, with columns added for the corresponding unique segment code and number
- segdata: data frame with one row for each segment, and columns with relevant data (see `das_effort` for specifics)

`das_chop_equallength` *Chop DAS data - equal length*

Description

Chop DAS data into approximately equal-length effort segments, averaging conditions by segment

Usage

```
das_chop_equallength(x, ...)

## S3 method for class 'data.frame'
das_chop_equallength(x, ...)

## S3 method for class 'das_df'
das_chop_equallength(
  x,
  conditions,
  seg.km,
  randpicks.load = NULL,
  distance.method = NULL,
  num.cores = NULL,
  ...
)
```

Arguments

x	an object of class das_df, or a data frame that can be coerced to class das_df. This data must be filtered for 'continuous effort sections; see the Details section below
...	ignored
conditions	see das_effort
seg.km	numeric; target segment length in kilometers
randpicks.load	character, data frame, or NULL. If character, must be filename of past randpicks output to load and use (passed to file argument of read.csv). If data frame, randpicks values will be extracted from the data frame. If NULL, new randpicks values will be generated by the function
distance.method	character; see das_effort. Default is NULL since these distances should have already been calculated
num.cores	see das_effort

Details

WARNING - do not call this function directly! It is exported for documentation purposes, but is intended for internal package use only.

This function is intended to be called by das_effort when the "equallength" method is specified. Thus, x must be filtered for events (rows) where either the 'OnEffort' column is TRUE or the 'Event' column is "E"; see das_effort for more details. This function chops each continuous effort section (henceforth 'effort sections') in x into modeling segments (henceforth 'segments') of equal length. Each effort section runs from an "R" event to its corresponding "E" event. After chopping, das_segdata is called to get relevant segdata information for each segment.

When chopping the effort sections in segments of length seg.km, there are several possible scenarios:

- The extra length remaining after chopping is greater than or equal to half of the target segment length (i.e. `>= 0.5*seg.km`): the extra length is assigned to a random portion of the effort section as its own segment (see Fig. 1a)
- The extra length remaining after chopping is less than half of the target segment length (i.e. `< 0.5*seg.km`): the extra length is added to one of the (randomly selected) equal-length segments (see Fig. 1b)
- The length of the effort section is less than or equal to the target segment length: the entire segment becomes a segment (see Fig. 1c)
- The length of the effort section is zero: a segment of length zero. If there are more than two events (the "B"/R" and "E" events), the function throws a warning

Therefore, the length of each segment is constrained to be between one half and one and one half of `seg.km` (i.e. `0.5*seg.km` <= segment length `>=1.5*seg.km`), and the central tendency is approximately equal to the target segment length. The only exception is when a continuous effort section is less than one half of the target segment length (i.e. `< 0.5*seg.km`; see Fig. 1c).

Note the PDF with Figs. 1a - 1c is included in the package, and can be found at: `system.file("DAS_chop_equal_figures.p`
`= "swfscDAS")`

'Randpicks' is a record of the random assignments that were made when chopping the effort sections into segments, and can be saved to allow users to recreate the same random allocation of extra km when chopping. The randpicks returned by this function is a data frame with two columns: the number of the effort section and the randpick value. Users should save the randpicks output to a CSV file, which then can be specified using the `randpicks.load` argument to recreate the same effort segments from `x` (i.e., using the same DAS data) in the future. Note that when saving with `write.csv`, users must specify `row.names = FALSE` so that the CSV file only has two columns. For an example randpicks file, see `system.file("das_sample_randpicks.csv",package = "swfscDAS")`

If the column `dist_from_prev` does not exist, the distance between subsequent events is calculated as described in `das_effort`

Value

List of three data frames:

- `x`, with columns added for the corresponding unique segment code and number
- segdata: data frame with one row for each segment, and columns with relevant data (see `das_effort` for specifics)
- randpicks: data frame with record of length allocations (see Details section above)

`das_chop_section` *Chop DAS data - section*

Description

Chop DAS data into effort segments by continuous effort section

Usage

```
das_chop_section(x, ...)

## S3 method for class 'data.frame'
das_chop_section(x, ...)

## S3 method for class 'das_df'
das_chop_section(x, conditions, distance.method = NULL, num.cores = NULL, ...)
```

Arguments

x	an object of class das_df, or a data frame that can be coerced to class das_df This data must be filtered for 'OnEffort' events; see the Details section below
...	ignored
conditions	see das_effort
distance.method	character; see das_effort. Default is NULL since these distances should have already been calculated
num.cores	see das_effort

Details

WARNING - do not call this function directly! It is exported for documentation purposes, but is intended for internal package use only.

This function is simply a wrapper for das_chop_equallength. It calls das_chop_equallength, with seg.km set to a value larger than the longest continuous effort section in x. Thus, the effort is 'chopped' into the continuous effort sections and then summarized.

See the Examples section for an example where the two methods give the same output. Note that the longest continuous effort section in the sample data is ~22km.

Value

See das_chop_equallength. The randpicks values will all be NA

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
y.proc <- das_process(y)

y.eff1 <- das_effort(y.proc, method = "equallength", seg.km = 25, num.cores = 1)
y.eff2 <- das_effort(y.proc, method = "section", num.cores = 1)

all.equal(y.eff1, y.eff2)
```

das_comments *Extract comments from DAS data*

Description

Extract comments from DAS data

Usage

```
das_comments(x)

## S3 method for class 'data.frame'
das_comments(x)

## S3 method for class 'das_df'
das_comments(x)

## S3 method for class 'das_dfr'
das_comments(x)
```

Arguments

x	an object of class das_dfr or das_df, or a data frame that can be coerced to a das_dfr object

Details

This function recreates the comment strings by pasting the Data# columns back together for the C events (comments). See the examples section for how to search for comments with certain phrases

Value

x, filtered for C events and with the added column comment_str containing the concatenated comment strings

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
y.proc <- das_process(y)

das_comments(y.proc)

# Extract all comments containing "record" - could also use stringr pacakge
y.comm <- das_comments(y.proc)
y.comm[grepl("record", y.comm$comment_str, ignore.case = TRUE), ]

# Join comments with processed data
dplyr::left_join(y.proc, y.comm[, c("file_das", "line_num", "comment_str")],
```

```
                    by = c("file_das", "line_num"))
```

`das_df-class` `das_df` *class*

Description

The `das_df` class is a subclass of `data.frame`, created to provide a concise and robust way to ensure that the input to downstream DAS processing functions, such as `das_sight`, adheres to certain requirements. Specifically, objects of class `das_df` are data frames with specific column names and classes, as detailed in the 'Properties of `das_df`' section. Objects of class `das_df` are created by `das_process` or `as_das_df`, and are intended to be passed directly to DAS processing functions such as `das_sight`.

Subsetting, say for a specific date or cruise number, or otherwise altering an object of class `das_df` will cause the object to drop its `das_df` class attribute. If this object is then passed to a DAS processing function such as `das_sight`, the function will try to coerce the object to a `das_df` object.

Properties of `das_df` objects

All values in the Event column must not be `NA`.

Objects of class `das_df` have a class attribute of `c("das_df","data.frame")`. In addition, they must have the following column names and classes:

Column name	*Column class*
Event	"character"
DateTime	c("POSIXct", "POSIXt")
Lat	"numeric"
Lon	"numeric"
OnEffort	"logical"
Cruise	"numeric"
Mode	"character"
EffType	"character"
Course	"numeric"
SpdKt	"numeric"
Bft	"numeric"
SwellHght	"numeric"
WindSpdKt	"numeric"
RainFog	"numeric"
HorizSun	"numeric"
VertSun	"numeric"
Glare	"logical"
Vis	"numeric"
ObsL	"character"
Rec	"character"
ObsR	"character"

ObsInd	"character"
Data1	"character"
Data2	"character"
Data3	"character"
Data4	"character"
Data5	"character"
Data6	"character"
Data7	"character"
Data8	"character"
Data9	"character"
Data10	"character"
Data11	"character"
Data12	"character"
EffortDot	"logical"
EventNum	"integer"
file_das	"character"
line_num	"integer"

See Also

as_das_df

`das_dfr-class` `das_dfr` *class*

Description

The das_dfr class is a subclass of data.frame, created to provide a concise and robust way to ensure that the input to das_process adheres to certain requirements. Specifically, objects of class das_dfr are data frames with specific column names and classes, as detailed in the 'Properties of das_dfr' section. Objects of class das_dfr are created by das_read or as_das_dfr, and are intended to be passed directly to das_process.

Subsetting or otherwise altering an object of class das_dfr will cause the object to drop its das_dfr class attribute. das_process will then try to coerce the object to a das_dfr object. It is **strongly** recommended to pass an object of class das_dfr to das_process before subsetting, e.g. for events from a certain date range.

Properties of das_dfr objects

Objects of class das_dfr have a class attribute of c("das_dfr","data.frame"). In addition, they must have the following column names and classes:

Column name	*Column class*
Event	"character"
EffortDot	"logical"
DateTime	c("POSIXct", "POSIXt")

Lat	"numeric"
Lon	"numeric"
Data1	"character"
Data2	"character"
Data3	"character"
Data4	"character"
Data5	"character"
Data6	"character"
Data7	"character"
Data8	"character"
Data9	"character"
Data10	"character"
Data11	"character"
Data12	"character"
EventNum	"integer"
file_das	"character"
line_num	"integer"

See Also

as_das_dfr

`das_effort` *Summarize DAS effort*

Description

Chop DAS data into effort segments

Usage

```
das_effort(x, ...)

## S3 method for class 'data.frame'
das_effort(x, ...)

## S3 method for class 'das_df'
das_effort(
  x,
  method = c("condition", "equallength", "section"),
  conditions = NULL,
  strata.files = NULL,
  distance.method = c("greatcircle", "lawofcosines", "haversine", "vincenty"),
  seg0.drop = FALSE,
  comment.drop = FALSE,
  event.touse = NULL,
```

```
  num.cores = NULL,
  ...
)
```

Arguments

x	an object of class das_df, or a data frame that can be coerced to class das_df
...	arguments passed to the specified chopping function, such as seg.km or seg.min.km
method	character; method to use to chop DAS data into effort segments Can be "condition", "equallength", "section", or any partial match thereof (case sensitive)
conditions	character vector of names of conditions to include in segdata output. These values must be column names from the output of das_process, e.g. 'Bft', 'SwellHght', etc. If method == "condition", then these also are the conditions which trigger segment chopping when they change. Only the following conditions can be used for chopping: 'Bft', 'SwellHght', 'RainFog', 'HorizSun', 'VertSun', 'Glare', 'Vis', 'Course', 'SpdKt'
strata.files	list of path(s) of the CSV file(s) with points defining each stratum. The CSV files must contain headers and be a closed polygon. The list should be named; see the Details section. If NULL (the default), then no effort segments are not classified by strata.
distance.method	character; method to use to calculate distance between lat/lon coordinates. Can be "greatcircle", "lawofcosines", "haversine", "vincenty", or any partial match thereof (case sensitive). Default is "greatcircle"
seg0.drop	logical; flag indicating whether or not to drop segments of length 0 that contain no sighting (S, K, M, G, t) events. Default is FALSE
comment.drop	logical; flag indicating if comments ("C" events) should be ignored (i.e. position information should not be used) when segment chopping. Default is FALSE
event.touse	character vector of events to use to determine segment lengths; overrides comment.drop. If NULL (the default), then all on effort events are used. If used, this argument must include at least R, E, S, and A events, and cannot include ? or 1:8 events
num.cores	Number of CPUs to over which to distribute computations. Defaults to NULL, which uses one fewer than the number of cores reported by detectCores. Using 1 core likely will be faster for smaller datasets

Details

This is the top-level function for chopping processed DAS data into modeling segments (henceforth 'segments'), and assigning sightings and related information (e.g., weather conditions) to each segment. This function returns data frames with all relevant information for the effort segments and associated sightings ('segdata' and 'sightinfo', respectively). Before chopping, the DAS data is filtered for events (rows) where either the 'OnEffort' column is TRUE or the 'Event' column "E". In other words, the data is filtered for continuous effort sections (henceforth 'effort sections'), where effort sections run from "R" to "E" events (inclusive), and then passed to the chopping function specified using method. Note that while B events immediately preceding an R are on effort, they are ignored during effort chopping. In addition, all on effort events (other than ? and numeric events) with NA DateTime, Lat, or Lon values are verbosely removed.

If strata.files is not NULL, then the effort lines will be split by the user-provided stratum (strata). In this case, a column 'stratum' will be added to the end of the segdata data frame with the user-provided name of the stratum that the segment was in, or NA if the segment was not in any of the strata. If no name was provided for the stratum in strata.files, then the value will be "Stratum#", where "#" is the index of the applicable stratum in strata.files. While the user can provide as many strata as they want, these strata can share boundaries but they cannot overlap. See das_effort_strata for more details.

The following chopping methods are currently available: "condition", "equallength", and "section. When using the "condition" method, effort sections are chopped into segments every time a condition changes, thereby ensuring that the conditions are consistent across the entire segment. See das_chop_condition for more details about this method, including arguments that must be passed to it via the argument ...

The "equallength" method consists of chopping effort sections into equal-length segments of length seg.km, and doing a weighted average of the conditions for the length of that segment. See das_chop_equallength for more details about this method, including arguments that must be passed to it via the argument ...

The "section" method involves 'chopping' the effort into continuous effort sections, i.e. each continuous effort section is a single effort segment. See das_chop_section for more details about this method.

The distance between the lat/lon points of subsequent events is calculated using the method specified in distance.method. If "greatcircle", distance_greatcircle is used, while distance is used otherwise. See das_sight for how the sightings are processed.

The sightinfo data frame includes the column 'included', which is used in das_effort_sight when summarizing the number of sightings and animals for selected species. das_effort_sight is a separate function to allow users to personalize the included values as desired for their analysis. By default, i.e. in the output of this function, 'included' is TRUE if: the sighting was made when on effort, by a standard observer (see das_sight), and in a Beaufort sea state less than or equal to five.

Value

List of three data frames:

- segdata: one row for every segment, and columns for information including unique segment number (segnum), the corresponding effort section (section_id), the segment index within the corresponding effort section (section_sub_id), the starting and ending line of the segment in the DAS file (stlin, endlin), start/end/midpoint coordinates(lat1/lon1, lat2/lon2, and mlat/mlon, respectively), the start/end/midpoint date/time of the segment (DateTime1, DateTime2, and mDateTime, respectively; mDateTime is the average of DateTime1 and DateTime2), segment length (dist), conditions (e.g. Beaufort), and, if applicable, stratum (InStratumName).
- sightinfo: details for all sightings in x, including: the unique segment number it is associated with, segment mid points (lat/lon), the 'included' column described in the 'Details' section, and the output information described in das_sight for return.format is "default"
- randpicks: see das_chop_equallength; NULL if using "condition" method

See Also

Internal functions called by das_effort: das_chop_condition, das_chop_equallength, das_chop_section, das_segdata

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
y.proc <- das_process(y)

# Using "condition" method
das_effort(
  y.proc, method = "condition", conditions = c("Bft", "SwellHght", "Vis"),
  seg.min.km = 0.05, num.cores = 1
)

# Using "section" method
das_effort(y.proc, method = "section", num.cores = 1)


# Using "equallength" method
y.rand <- system.file("das_sample_randpicks.csv", package = "swfscDAS")
das_effort(
  y.proc, method = "equallength", seg.km = 10, randpicks.load = y.rand,
  num.cores = 1
)

# Using "section" method and chop by strata
stratum.file <- system.file("das_sample_stratum.csv", package = "swfscDAS")
das_effort(
  y.proc, method = "section", strata.files = list(Poly1 = stratum.file),
  num.cores = 1
)
```

---

das_effort_sight *Summarize DAS sightings by effort segment*

Description

Summarize number of sightings and animals for selected species by segment

Usage

```
das_effort_sight(
  x.list,
  sp.codes,
  sp.events = c("S", "G", "K", "M", "t", "p"),
  gs.columns = c("GsSpBest", "GsSpLow", "GsSpHigh")
)
```

Arguments

x.list	output of das_effort; a list of three data frames named 'segdata', 'sightinfo', and 'randpicks', respectively
sp.codes	character; species code(s) to include in segdata output. These must exactly match the species codes in the data, such as including leading zeros
sp.events	character; event code(s) to include in the sightinfo output. This argument supersedes the 'included' value when determining whether a sighting is included in the segment summaries. Must be one or more of: "S", "K", "M", "G", "t", "p" (case-sensitive). The default is that all of these event codes are kept
gs.columns	character; the column(s) to use to get the group size values that will be summarized in the segdata output. Must be one or more of 'GsSpBest', 'GsSpLow', and 'GsSpBest' (case-sensitive). See Details section for more information

Details

This function takes the output of das_effort and adds columns for the number of sightings (nSI) and number of animals (ANI) for selected species (selected via sp.codes) for each segment to the segdata element of x.list. However, only sightings with an included value of TRUE (included is a column in sightinfo) are included in the summaries. Having this step separate from das_effort allows users to personalize the included values as desired for their analysis.

The ANI columns are the sum of the 'GsSp...' column(s) from das_sight specified using gs.columns. If gs.columns specifies more than one column, then the secondary columns will only be used if the values for the previous columns are NA. For instance, if gs.columns = c('GsSpBest','GsSpLow'), then for each row in sightinfo, the value from GsSpLow will be used only if the value from GsSpBest is NA

Value

A list, identical to x.list except for 1) the nSI and ANI columns added to x.list$segdata, one each for each element of sp.codes, and 2) the 'included' column of x.list$sightinfo, which has been set as FALSE for sightings of species not listed in sp.codes. Thus, the 'included' column in the output accurately reflects the sightings that were included in the effort segment summaries

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
y.proc <- das_process(y)
y.eff.cond <- das_effort(
  y.proc, method = "condition", conditions = "Bft", seg.min.km = 0.05,
  num.cores = 1
)

das_effort_sight(y.eff.cond, sp.codes = c("013", "076", "DC"), sp.events = c("S", "t"))
```

`das_effort_strata` *Split effort by strata*

Description

Split DAS effort where it intersects with a stratum boundary

Usage

```
das_effort_strata(x, ...)

## S3 method for class 'data.frame'
das_effort_strata(x, ...)

## S3 method for class 'das_df'
das_effort_strata(x, strata.files, ...)
```

Arguments

`x`	an object of class `das_df`, or a data frame that can be coerced to class `das_df`
`...`	ignored
`strata.files`	list of path(s) of the stratum CSV file(s); see `das_effort`

Details

This function should only be called by `das_effort`, i.e. it should not be called by users in their personal scripts. Practically speaking, this functions splits the effort line wherever it crosses a stratum line. This point of intersection is interpolated; specifically, it is determined using `st_intersection`. Thus, any effort will be first split at these effort-stratum boundary intersection points, and then using the specified method (e.g. condition).

Value

The data frame x, with 1) columns added that indicate a) if the point was in a particular stratum (see `das_intersects_strata`), and b) the index of the stratum in `strata.files` (column name 'stratum'; 0 if the point intersects with no strata), and 2) two rows added for each strata crossing that occurs between something other than an E and R. These rows are necessary because of how `das_effort` processes effort. The added rows are the same as the event previous to the strata crossing, except:

- They have the event code "strataE" and "strataR", respectively
- Their coordinates are the coordinates of the intersection of the effort line and the stratum boundary
- Their 'idx_eff' values are plus 0.4 and 0.5, respectively
- The second added row has the same stratum info as the point immediately after the stratum boundary crossing

das_format_pdf *DAS format requirements*

Description

Save the PDF document describing the DAS format required by swfscDAS to a specified file

Usage

```
das_format_pdf(file, ...)
```

Arguments

file	character, the name of the file where the PDF will be saved
...	passed on to file.copy; might included named argument overwrite

Details

A wrapper function for file.copy. This function saves the PDF document describing the DAS data format requirements by copying the PDF document located at system.file("DAS_Format.pdf",package = "swfscDAS") to file

This file can also be downloaded from https://github.com/smwoodman/swfscDAS/blob/master/inst/DAS_Format.pdf

Value

output of file.copy; TRUE if writing of file was successful, and FALSE otherwise

Examples

```
das_format_pdf(file.path(tempdir(), "DAS_Format.pdf"), overwrite = FALSE)
```

das_intersects_strata *DAS strata - points*

Description

Determine if swfscDAS outputs intersect with strata polygons

Usage

```
das_intersects_strata(x, ...)

## S3 method for class 'list'
das_intersects_strata(x, strata.files, ...)

## S3 method for class 'data.frame'
das_intersects_strata(
  x,
  strata.files,
  x.lon = "Lon",
  x.lat = "Lat",
  strata.which = FALSE,
  ...
)
```

Arguments

x	a data frame (such as an object of class das_df) or a list. If x is a list, then it must be the output of das_effort or das_effort_sight. If x is a data frame, the user must also specify the coordinate columns of x using x.lon and x.lat
...	ignored
strata.files	list of path(s) of the CSV file(s) with points defining each stratum. The CSV files must contain headers and be a closed polygon. The list may be named; see 'Value' section for how these names are used
x.lon	character; name of the longitude column of x. Ignored if x is a list; default is "Lon"
x.lat	character; name of the latitude column of x. Ignored if x is a list; default is "Lat"
strata.which	logical; indicates if the numeric column 'strata_which' should be included in the output data frame. Ignored if x is a list; default is FALSE. See 'Value' section for more details

Details

Assigns DAS event points or segment midpoints to strata polygons using st_intersects.

If x is a list, then 1) it must be the output of das_effort or das_effort_sight and 2) the segment midpoints (column names mlon and mlat, respectively) are the points checked if they intersect with each provided stratum. If x is a data frame, then the user must provide the columns that specify the point coordinates to check.

x should not be an object of class das_dfr, or an object of class das_df created with add.dtll.sight = FALSE, because the ? and numeric event codes will have NA latitude and longitude values.

Value

If x is a data frame, then logical columns are added to x indicating if each point intersected with the corresponding stratum polygon. The names of these columns are the names of strata.files;

the element(s) of `strata.files` will have the name InPoly#, where '#' is the index of that stratum polygon in `strata.files`. If `strata.which`, then the column 'strata_which' is added to the end of the data frame. This column contains either a 0 if the point intersects with no strata or 2) a numeric indicating the index (in `strata.files`) of the (first) strata polygon that the point intersects with.

Otherwise, i.e. if `x` is a list and thus the output of one of the effort functions, then the stratum columns are added to both the segdata and sightinfo data frames. However, note that the columns added to the sightinfo data frame still indicate whether or not the segment midpoint was in the corresponding stratum, rather than the sighting point itself.

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
y.proc <- das_process(y)
y.eff <- das_effort(y.proc, method = "section", num.cores = 1)

stratum.file <- system.file("das_sample_stratum.csv", package = "swfscDAS")
das_intersects_strata(y.eff, list(InPoly = stratum.file), x.lon = "Lon", x.lat = "Lat")

das_intersects_strata(y.proc, list(stratum.file))

# Visualize effort midpoints and stratum polygon
require(sf)
y.eff.strata <- das_intersects_strata(y.eff, list(InPoly = stratum.file))
segdata <- st_as_sf(y.eff.strata$segdata, coords = c("mlon", "mlat"), crs = 4326)

# Make stratum polygon
stratum.df <- read.csv(stratum.file)
stratum.sfc <- st_sfc(
  st_polygon(list(matrix(c(stratum.df$Lon, stratum.df$Lat), ncol = 2))),
  crs = 4326
)

plot(segdata["InPoly"], axes = TRUE, reset = FALSE,
     xlim = c(-137, -142.5), ylim = c(42, 47))
plot(stratum.sfc, add = TRUE)
```

`das_process` *Process DAS data*

Description

Process DAS data (the output of `das_read`), including extracting state and condition information for each DAS event

Usage

```
das_process(x, ...)
```

```
## S3 method for class 'character'
das_process(x, ...)

## S3 method for class 'data.frame'
das_process(x, ...)

## S3 method for class 'das_dfr'
das_process(
  x,
  days.gap = 20,
  reset.event = TRUE,
  reset.effort = TRUE,
  reset.day = TRUE,
  add.dtll.sight = TRUE,
  ...
)
```

Arguments

x	an object of class das_dfr, an object that can be coerced to class das_dfr, or a character (filepath) which is first passed to das_read
...	passed to das_read if x is a character. Otherwise ignored
days.gap	numeric of length 1; default is 20. Time gap (in days) used to identify a new cruise in concatenated DAS files, and thus also when state/condition information (cruise number, weather, Bft, Mode, etc) is reset
reset.event	logical; default is TRUE. Indicates if state/condition information (weather, Bft, Mode, etc) should be reset to NA if there is an applicable event with an NA for that state/condition
reset.effort	logical; default is TRUE. Indicates if state/condition information should be reset to NA when beginning a new continuous effort section. See Details section
reset.day	logical; default is TRUE. Indicates if state/condition information should be reset to NA at the beginning of each day. This argument should only be set to FALSE for comparison with older methods, such as REPORT
add.dtll.sight	logical indicating if the DateTime (dt) and latitude and longitude (ll) columns should be added to the sighting events (?, 1, 2, 3, 4, 5, 6, 7, and 8) from the corresponding (immediately preceding) A event

Details

If x is a character, it is assumed to be a filepath and first passed to das_read. This output is then passed to das_process.

DAS data is event-based, meaning most events indicate when a state or weather condition changes. For instance, a 'V' event indicates when one or more sea state viewing conditions (such as Beaufort sea state) change, and these conditions are the same for subsequent events until the next 'V' event. For each state/condition: a new column is created, the state/condition information is extracted from relevant events, and extracted information is propagated to appropriate subsequent rows (events).

Thus, each row in the output data frame contains all pertinent state/condition information for that row.

The following assumptions/decisions are made during processing:

- Event codes are expected to be one of the following: #, *, ?, 1, 2, 3, 4, 5, 6, 7, 8, A, B, C, E, F, k, K, M, N, P, Q, r, R, s, S, t, V, W, g, G, p, X, Y, Z
- All '#' events (deleted events) are removed
- r events are converted to R events with non-standard effort; see `das_format_pdf` for more details
- An event is considered 'on effort' if it is 1) an R event, 2) a B event immediately preceding an R event, or 3) between corresponding R and E events (not including the E event). The 'EffortDot' column is not used when determining on effort data. Note that effort is reset to 'off effort' at the beginning of a new day.
- All state/condition information is reset at the beginning of each cruise. New cruises are identified using `days.gap`.
- All state/condition information relating to B, R, P, V, N, and W events are reset every time there is a BR event sequence if `reset.effort == TRUE`, because in WinCruz a BR event sequence should always be a BRPVNW event sequence. An event sequence means that all of the events have the same Lat/Lon/DateTime info, and thus previous values for conditions set during the event sequence should not carry over to any part of the event sequence.
- 'OffsetGMT' is converted to an integer. Values are expected to be consistent within a day for each cruise, so events will have an OffsetGMT value if there is any B event with the offset data on the same day, whether that event is before or after the B event. Thus, if any date/cruise combinations have multiple OffsetGMT values in the data, then a warning message will be printed and the OffsetGMT values will be all NA (for the entire output).
- 'Mode' is capitalized, and 'Mode' values of `NA` are assigned a value of "C"
- 'EffType' is capitalized, and values of `NA` are assigned a value of "S"
- 'ESWsides' represents the number of sides being searched during that effort section - a value of `NA` (for compatibility with older data) or "F" means 2 sides are being searched, and a value of "H" means 1 side is being searched. ESWsides will be `NA` for values that are not one of "F", `NA`, or "H"
- 'Glare': `TRUE` if 'HorizSun' is 11, 12 or 1 and 'VertSun' is 2 or 3, or if 'HorizSun' is 12 and 'VertSun' is 1; `NA` if 'HorizSun' or 'VertSun' is `NA`; otherwise `FALSE`
- Missing values are `NA` rather than `-1`

Value

A `das_df` object, which is also a data frame. It consists of the input data frame, i.e. the output of `das_read`, with the following columns added:

State/condition	*Column name*	*Data source*
On/off effort	OnEffort	B/R and E events
Cruise number	Cruise	Event: B; Column: Data1
Effort mode	Mode	Event: B; Column: Data2
GMT offset of DateTime data	OffsetGMT	Event: B; Column: Data3

Effort type	EffType	Event: R; Column: Data1
Number of sides with observer	ESWSide	Event: R; Column: Data2
Course (ship direction)	Course	Event: N; Column: Data1
Speed (ship speed, knots)	SpdKt	Event: N; Column: Data2
Beaufort sea state	Bft	Event: V; Column: Data1
Swell height (ft)	SwellHght	Event: V; Column: Data2
Wind speed (knots)	WindSpdKt	Event: V; Column: Data5
Rain/fog/haze code	RainFog	Event: W; Column: Data1
Horizontal sun (clock system)	HorizSun	Event: W; Column: Data2
Vertical sun (clock system)	VertSun	Event: W; Column: Data3
Glare	Glare	HorizSun and VertSun
Visibility (nm)	Vis	Event: W; Column: Data5
Left observer	ObsL	Event: P; Column: Data1
Data recorder	Rec	Event: P; Column: Data2
Right observer	ObsR	Event: P; Column: Data3
Independent observer	ObsInd	Event: P; Column: Data4

OffsetGMT represents the difference in hours between the DateTime data (which should be in local time) and GMT (i.e., UTC).

Internal warning messages are printed with row numbers of the input file (NOT of the output data frame) of unexpected event codes and r events, as well as if there is are potential issues with the number and/or order of R and E events

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
das_process(y)

y.read <- das_read(y)
das_process(y.read)
das_process(y.read, reset.effort = FALSE)
```

`das_read` *Read DAS file(s)*

Description

Read one or more fixed-width DAS text file(s) generated by WinCruz into a data frame, where each line is data for a specific event

Usage

```
das_read(file, skip = 0, ...)
```

Arguments

`file`	filename(s) of one or more DAS files
`skip`	integer; see `read_fwf`. Default is 0
`...`	ignored

Details

Reads/parses DAS data into columns of a data frame. If `file` contains multiple filenames, then the individual data frames will be concatenated.

The provided DAS file must adhere to the following column number and format specifications:

Item	*Columns*	*Format*
Event number	1-3	
Event	4	
Effort dot	5	
Time	6-11	HHMMSS or HHMM
Date	13-18	MMDDYY
Latitude	20-28	NDD:MM.MM
Longitude	30-39	WDDD:MM.MM
Data1	40-44	
Data2	45-49	
Data3	50-54	
Data4	55-59	
Data5	60-64	
Data6	65-69	
Data7	70-74	
Data8	75-79	
Data9	80-84	
Data10	85-89	
Data11	90-94	
Data12	95+	

See `das_format_pdf` for more information about DAS format requirements, and note that 'Data#' columns may be referred to as 'Field#' columns in other documentation.

Value

A `das_dfr` object, which is also a data frame, with DAS data read into columns. The data are read into the data frame as characters as described in 'Details', with the following exceptions:

Name	*Class*	*Details*
EffortDot	logical	`TRUE` if "." was present, and `FALSE` otherwise
DateTime	POSIXct	combination of 'Date' and 'Time' columns
Lat	numeric	'Latitude' column converted to decimal degrees in range [-90, 90]
Lon	numeric	'Longitude' column converted to decimal degrees in range [-180, 180]
Data#	character	leading/trailing whitespace trimmed for non-comment events (i.e. where 'Event' is not "C")

EventNum	character	leading/trailing whitespace trimmed; left as character for some project-specific codes
file_das	character	base filename, extracted from the `file` argument
line_num	integer	line number of each data row

DateTime values have a (meaningless) time zone value of "UTC". See the OffsetGMT column from `das_process` for relevant time zone information

Warnings are printed if any unexpected events have `NA` DateTime/Lat/Lon values, or if any Lat/Lon values cannot be converted to numeric values. Events that are 'expected' to have `NA` DateTime/Lat/Lon values are: C, ?, 1, 2, 3, 4, 5, 6, 7, 8

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
das_read(y)
```

---

das_segdata *Summarize DAS data for a continuous effort section*

Description

Summarize DAS effort data by effort segment, while averaging or getting the max for each condition

Usage

```
das_segdata(x, ...)

## S3 method for class 'data.frame'
das_segdata(x, ...)

## S3 method for class 'das_df'
das_segdata(
  x,
  conditions,
  segdata.method = c("avg", "maxdist"),
  seg.lengths,
  section.id,
  ...
)
```

Arguments

`x`	an object of class `das_df`, or a data frame that can be coerced to class `das_df` Must contain a single continuous effort section of DAS data; see the Details section below
`...`	ignored

conditions	see das_effort, or see Details section for more information
segdata.method	character; either avg" or "maxdist". See Details section for more information
seg.lengths	numeric; length of the modeling segments into which x will be chopped
section.id	numeric; the ID of x (the current continuous effort section)

Details

WARNING - do not call this function directly! It is exported for documentation purposes, but is intended for internal package use only.

This function was designed to be called by one of the das_chop_ functions, e.g. das_chop_equallength, and thus users should avoid calling it themselves. It loops through the events in x, chopping x into modeling segments while calculating and storing relevant information for each segment. Because x is a continuous effort section, it must begin with a "B" or "R" event and end with the corresponding "E" event.

For each segment, this function reports the segment number, segment ID, cruise number, the start/end/mid coordinates (lat/lon), start/end/mid date/times (DateTime), segment length, year, month, day, mid-point time, mode, effort type, effective strip width sides (number of sides searched), and average conditions (which are specified by conditions). The segment ID is designated as section.id _ index of the modeling segment. Thus, if section.id is 1, then the segment ID for the second segment from x is "1_2". The start/end coordinates and date/times are interpolated as needed, e.g. when using the 'equallength' method.

When segdata.method is "avg", the condition values are calculated as a weighted average by distance. The reported value for logical columns (e.g. Glare) is the percentage (in decimals) of the segment in which that condition was TRUE. For character columns, the reported value for each segment is the unique value(s) present in the segment, with NAs omitted, pasted together via paste(...,collapse = "; "). When segdata.method is "maxdist", the reported values are, for each condition, the value recorded for the longest distance during that segment (with NAs omitted).

Cruise number, mode, effort type, sides searched, and file name are also included in the segdata output. These values (excluding NAs) must be consistent across the entire effort section, and thus across all segments in x; a warning is printed if there are any inconsistencies

bearing and destination are used to calculate the segment start, mid, and end points, with method = "vincenty".

Value

Data frame with the segdata information described in Details and in das_effort

---

das_sight *DAS sightings*

Description

Extract sightings and associated information from processed DAS data

Usage

```
das_sight(x, ...)

## S3 method for class 'data.frame'
das_sight(x, ...)

## S3 method for class 'das_df'
das_sight(
  x,
  return.format = c("default", "wide", "complete"),
  return.events = c("S", "K", "M", "G", "s", "k", "m", "g", "t", "p", "F"),
  ...
)
```

Arguments

x	an object of class das_df, or a data frame that can be coerced to class das_df
...	ignored
return.format	character; can be one of "default", "wide", "complete", or any partial match thereof (case sensitive). Formats described below
return.events	character; event codes included in the output. Must be one or more of: "S", "K", "M", "G", "s", "k", "m", "g", "t", "p", "F" (case-sensitive). The default is all of these event codes

Details

DAS events contain specific information in the 'Data#' columns, with the information depending on the event code for that row. The output data frame contains columns with this specific information extracted to dedicated columns as described below. This function recognizes the following types of sightings: marine mammal sightings (event codes "S", "K", or "M"), marine mammal resights (codes "s", "k", "m"), marine mammal subgroup sightings (code "G"), marine mammal subgroup resights (code "g"), turtle sightings (code "t"), pinniped sightings (code "p") and fishing vessel sightings (code "F"). Warnings are printed if all S, K, M, and G events (and only these events) are not followed by an A event and at least one numeric event. See das_format_pdf for more information about events and event formats. Of specific note - sperm whale sightings (species code 046) often contain additional estimates recorded as "C" events immediately following the S, A, and numeric events. Because these estimates are recorded as"C" events, they are NOT included in the das_sight calculations or output for any return.format

The return.events argument simply provides a shortcut for filtering the output of das_sight by event codes

Abbreviations used in output column names: Gs = group size, Sp = species, Nm = nautical mile, Perc = percentage, Prob = probable, GsSchool = school-level group size info

This function makes the following assumptions, and alterations to the raw DAS data:

- "A" events immediately following an S/K/M/G event have the same sighting number (Data1 value) as the S/K/M/G event

- The 'nSp' column is equivalent to the number of non-NA values across the 'Data5', 'Data6', 'Data7', and 'Data8' columns for the pertinent "A" event
- The following data are coerced to a numeric using as.numeric: Bearing, Reticle, DistNm, Cue, Method, species percentages, and group sizes (including for t, p, and F events). Note that if there are any formatting errors and these data are not numeric, the function will likely print a warning message
- The values for the following columns are capitalized using toupper: 'Birds', 'Photos', 'CalibSchool', 'PhotosAerial', 'Biopsy', 'TurtleAge', and 'TurtleCapt'

Value

Data frame with 1) the columns from x, excluding the 'Data#' columns, and 2) columns with sighting information extracted from 'Data#' columns. See das_format_pdf for more information the sighting information. If return.format is "default", then there is one row for each species of each sighting event; if return.format is "wide", then there is one row for each sighting event; if return.format is "complete", then there is one row for every group size estimate for each sighting event (excluding sperm whale "C" events - see the Details section).

The format-specific columns are described in their respective sections. The following sighting information columns are included in all return formats:

Sighting information	*Column name*	*Notes*
Sighting number	SightNo	Character
Subgroup code	Subgroup	Character
Daily sighting number	SightNoDaily	See below
Observer that made the sighting	Obs	
Standard observer	ObsStd	Logical; TRUE if Obs is one of ObsL, Rec or ObsR, and FALSE o
Bearing to the sighting	Bearing	Numeric; degrees, expected range 0 to 360
Number of reticle marks	Reticle	Numeric
Distance (nautical miles)	DistNm	Numeric
Sighting cue	Cue	
Sighting method	Method	
Photos of school?	Photos	
Birds present with school?	Birds	
Calibration school?	CalibSchool	
Aerial photos taken?	PhotosAerial	
Biopsy taken?	Biopsy	
Probable sighting	Prob	Logical indicating if sighting has associated ? event; NA for non-
Number of species in sighting	nSp	NA for non-S/K/M/G events
Mixed species sighting	Mixed	Logical; TRUE if nSp > 1
Group size of school - best estimate	GsSchoolBest	See below
Group size of school - high estimate	GsSchoolHigh	See below
Group size of school - low estimate	GsSchoolLow	See below
Course (true heading) of school at resight	CourseSchool	NA for non-s/k/m events
Presence of associated JFR	TurtleJFR	NA for non-"t" events; JFR = jellyfish, floating debris, or red tide
Estimated turtle maturity	TurtleAge	NA for non-"t" events
Perpendicular distance (km) to sighting	PerpDistKm	Calculated via (abs(sin(Bearing*pi/180) * DistNm) * 1.85

SightNoDaily is a running count of the number of S/K/M/G sightings that occurred on each day.

It is formatted as 'YYYYMMDD'_'running count', e.g. "20050101_1". The GsSchoolBest, GsSchoolHigh, and GsSchoolLow columns are either: 1) the arithmetic mean across observer estimates for the "default" and "wide" formats, or 2) the individual observer estimates for the "complete" format. Note that for non-"complete" formats, `na.rm = TRUE` is used when calculating the mean, and thus blank elements of estimates (but not the whole incomplete estimate) are ignored. To convert the perpendicular distance back to nautical miles, one would divide PerpDistKm by 1.852

The "default" format output

This output data frame contains 'long' sighting data, meaning there is one row for each species of each sighting event. The GsSp... columns are calculated as follows: for each species and for each observer estimate, the best/high/low school size estimate is multiplied by the applicable species percent estimate. The values are grouped by species and then averaged to get single GsSpBest, GsSpHigh, and GsSpLow values for each species. (using `mean` with `na.rm = TRUE`)

Sighting information columns/formats present specifically in the "default" format output:

Sighting information	*Column name*	*Notes*
Species code	SpCode	Boat type or mammal, turtle, or pinniped species codes
Probable species code	SpCodeProb	Probable mammal species codes; `NA` if none or not applicable
Group size of species - best estimate	GsSpBest	Arithmetic mean, across observer estimates, of the product(s) of GsSc
Group size of species - high estimate	GsSpHigh	Arithmetic mean, across observer estimates, of the product(s) of GsSc
Group size of species - low estimate	GsSpLow	Arithmetic mean, across observer estimates, of the product(s) of GsSc

The "wide" and "complete" format outputs

The "wide" and "complete" options have very similar columns in their output date frames. There are two main differences: 1) the "wide" format has one row for each sighting event, while the complete format has a row for every observer estimate for each sightings, and thus 2) in the "wide" format, all numeric information for which there are multiple observer estimates (school group size, species percentage, etc.) are averaged across estimated via an arithmetic mean (using `mean` with `na.rm = TRUE`)

With these formats, note that the species/type code and group size for turtle, pinniped, and boat sightings are in their own column

Sighting information columns present in the "wide" and "complete" format outputs:

Sighting information	*Column name*	*Notes*
Observer code - estimate	ObsEstimate	See below
Species 1 code	SpCode1	
Species 2 code	SpCode2	
Species 3 code	SpCode3	
Species 4 code	SpCode4	
Species 1 probable code	SpCodeProb1	Extracted from '?' event
Species 2 probable code	SpCodeProb2	Extracted from '?' event
Species 3 probable code	SpCodeProb3	Extracted from '?' event
Species 4 probable code	SpCodeProb4	Extracted from '?' event
Percentage of Sp 1 in school	SpPerc1	
Percentage of Sp 2 in school	SpPerc2	

Percentage of Sp 3 in school	SpPerc3	
Percentage of Sp 4 in school	SpPerc4	
Group size of species 1	GsSpBest1	Present in "wide" output only; see below
Group size of species 2	GsSpBest2	Present in "wide" output only; see below
Group size of species 3	GsSpBest3	Present in "wide" output only; see below
Group size of species 4	GsSpBest4	Present in "wide" output only; see below
Turtle species	TurtleSp	NA for non-"t" events
Turtle group size	TurtleGs	NA for non-"t" events
Was turtle captured?	TurtleCapt	NA for non-"t" events
Pinniped species	PinnipedSp	NA for non-"p" events
Pinniped group size	PinnipedGs	NA for non-"p" events
Boat or gear type	BoatType	NA for non-"F" events
Number of boats	BoatGs	NA for non-"F" events

ObsEstimate refers to the code of the observer that made the corresponding estimate. For the "wide" format, ObsEstimate is a list-column of all of the observer codes that provided an estimate. Also in the "wide" format, the GsSpBest# columns are the product of GsSchoolBest and the corresponding species percentage. These numbers, 1 to 4, correspond to the order of the data as it appears in the DAS file

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
y.proc <- das_process(y)

das_sight(y.proc)
das_sight(y.proc, return.format = "complete")
```

---

distance_greatcircle *Calculate great-circle distance*

Description

Calculate the great-circle distance between two lat/lon points

Usage

```
distance_greatcircle(lat1, lon1, lat2, lon2)
```

Arguments

lat1	numeric; starting latitude coordinate(s)
lon1	numeric; starting longitude coordinate(s)
lat2	numeric; ending latitude coordinate(s)
lon2	numeric; ending longitude coordinate(s)

Value

Distance in kilometers between lat1/lon1 and lat2/lon2

See Also

https://en.wikipedia.org/wiki/Great-circle_distance

`randpicks_convert` *Convert randpicks file*

Description

Convert randpicks file from segchopr format to swfscDAS format

Usage

```
randpicks_convert(x.randpicks, x.segdata, seg.km)
```

Arguments

`x.randpicks`	Data frame with two columns; randpick values formatted for segchopr that correspond to `x.segdata`
`x.segdata`	Data frame; segdata that corresponds to `x.randpicks`
`seg.km`	numeric; target segment length used when creating `x.segdata`

Details

Past DAS processing code (segchopr) only recorded the generated random values, whereas swfscDAS randpicks files contain one line for each continuous effort section. See `das_chop_equallength` for more details about the swfscDAS randpicks format. This function 'converts' a randpicks data frame generated by segchopr to a data frame that meets the swfscDAS randpicks format requirements

Value

Data frame with one line for each continuous effort section in `x.segdata`, and two columns: `effort_section` and `randpicks`

subsetting *Subsetting objects created using swfscDAS*

Description

Subsetting das_dfr or das_df objects

Usage

```
## S3 method for class 'das_dfr'
x[i, j, ..., drop = TRUE]

## S3 replacement method for class 'das_dfr'
x$name <- value

## S3 replacement method for class 'das_dfr'
x[i, j, ...] <- value

## S3 replacement method for class 'das_dfr'
x[[i]] <- value

## S3 method for class 'das_df'
x[i, j, ..., drop = TRUE]

## S3 replacement method for class 'das_df'
x$name <- value

## S3 replacement method for class 'das_df'
x[i, j, ...] <- value

## S3 replacement method for class 'das_df'
x[[i]] <- value
```

Arguments

x	object of class das_dfr or das_df
i, j, ...	elements to extract or replace, see [.data.frame
drop	logical, see [.data.frame
name	A literal character string or ..., see [.data.frame
value	A suitable replacement value, see [.data.frame

Details

When subsetting a das_dfr or das_df object, henceforth a das_ object, using any of the functions described in [.data.frame, then then the das_ class is simply dropped and the object is of class data.frame. This is because of the strict format requirements of das_ objects; it is likely that a

subsetted das_ object will not have the format required by subsequent swfscDAS functions, and thus it is safest to drop the das_ class. If a data frame is passed to downstream swfscDAS functions that require a das_ object, then they will attempt to coerce the object to the necessary das_ class See as_das_dfr and as_das_df for more details.

Examples

```
y <- system.file("das_sample.das", package = "swfscDAS")
y.read <- das_read(y)

# All return a data frame:
class(y.read[1:10, ])
class(y.read[, 1:10])

y.df <- y.read
y.df[, 1] <- "a"
class(y.df)

y.df <- y.read
y.df$Event <- "a"
class(y.df)

y.df <- y.read
y.df[["Event"]] <- "a"
class(y.df)
```

---

swfscAirDAS-internals *Internal functions for swfscAirDAS*

Description

These functions are exported only to be used internally by swfscAirDAS. They implement functionality that is used when processing both DAS and AirDAS data

Usage

```
.chop_condition_eff(i, call.x, call.conditions, call.seg.min.km, call.func1)

.chop_equallength_eff(
  i,
  call.x,
  call.conditions,
  call.seg.km,
  call.r.pos,
  call.func1
)

.process_num(init.val, das.df, col.name, event.curr, event.na)
```

```
.process_chr(init.val, das.df, col.name, event.curr, event.na)

.segdata_proc(
  das.df,
  conditions,
  segdata.method,
  seg.lengths,
  section.id,
  df.out1
)

.segdata_aggr(data.list, curr.df, idx, dist.perc)

.dist_from_prev(
  z,
  z.distance.method = c("greatcircle", "lawofcosines", "haversine", "vincenty")
)
```

Arguments

i	ignore
call.x	ignore
call.conditions	ignore
call.seg.min.km	ignore
call.func1	ignore
call.seg.km	ignore
call.r.pos	ignore
init.val	ignore
das.df	ignore
col.name	ignore
event.curr	ignore
event.na	ignore
conditions	ignore
segdata.method	ignore
seg.lengths	ignore
section.id	ignore
df.out1	ignore
data.list	ignore
curr.df	ignore
idx	ignore
dist.perc	ignore

`z` ignore

`z.distance.method` ignore

Index