

# Package ‘swfscDAS’

June 30, 2020

**Title** Southwest Fisheries Science Center Shipboard DAS Data Processing

**Version** 0.4.0

**Description** Process and summarize shipboard DAS data (data written to a .das file) collected using the Southwest Fisheries Science Center (SWFSC) 'WinCruz' program. This package is intended to standardize and streamline basic DAS data processing. A PDF with the DAS data format requirements is included in this package.

**URL** <https://smwoodman.github.io/swfscDAS>,  
<https://github.com/smwoodman/swfscDAS>

**BugReports** <https://github.com/smwoodman/swfscDAS/issues>

**Depends** R (>= 3.5.0)

**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

**LazyData** true

**RoxygenNote** 7.1.0

**VignetteBuilder** knitr

**NeedsCompilation** no

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

**Maintainer** Sam Woodman <[sam.woodman@noaa.gov](mailto:sam.woodman@noaa.gov)>

**Repository** CRAN

**Date/Publication** 2020-06-30 09:20:07 UTC

## R topics documented:

swfscDAS-package . . . . .	2
as_das_df . . . . .	3
as_das_dfr . . . . .	3

das_check . . . . .	4
das_comments . . . . .	6
das_df-class . . . . .	8
das_dfr-class . . . . .	9
das_effort . . . . .	10
das_effort_sight . . . . .	13
das_format_pdf . . . . .	14
das_intersects_strata . . . . .	15
das_process . . . . .	16
das_read . . . . .	19
das_sight . . . . .	22
distance_greatcircle . . . . .	25
randpicks_convert . . . . .	26
subsetting . . . . .	27
swfscAirDAS-internals . . . . .	28
<b>Index</b>	<b>31</b>

---

 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	<i>Coerce object to a das_df object</i>
-----------	---

---

### 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	<i>Coerce object to a das_dfr object</i>
------------	--

---

### 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	<i>Check DAS file</i>
-----------	-----------------------

---

**Description**

Check that DAS file has accepted values

**Usage**

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

## Arguments

file	filename(s) of one or more DAS files
skip	integer: see <a href="#">read_fwf</a> . 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, the 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 <a href="#">table</a> . Default is TRUE

## Details

Precursor to a more comprehensive DASCHECK program. 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.
- 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:

<i>Item</i>	<i>Event</i>	<i>Column</i>	<i>Requirement</i>
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)
Effective strip width sides	R	Data2	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 be either a whole number or have a de
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

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

- Add check for sequential date/time
- Add column with cruise number to output
- 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)
- Check that lat/lon values are within [-90, 90] and [-180, 180]

### 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 is also written to a text file

A warning is printed if any events are r events

### Examples

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

---

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 package
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:

<i>Column name</i>	<i>Column class</i>
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:

<i>Column name</i>	<i>Column class</i>
Event	"character"
EffortDot	"logical"
DateTime	<code>c("POSIXct", "POSIXt")</code>
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,
  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 <code>das_df</code> , or a data frame that can be coerced to class <code>das_df</code>
...	arguments passed to the specified chopping function, such as <code>seg.km</code> or <code>seg.min.km</code>
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 <code>segdata</code> output. These values must be column names from the output of <code>das_process</code> , e.g. 'Bft', 'Swell-Hght', etc. If <code>method == "condition"</code> , then these also are the conditions which trigger segment chopping when they change.
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 <code>comment.drop</code> . 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 <code>detectCores</code> . 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.

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, start/end/midpoint coordinates, and conditions (e.g. Beaufort)
- `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](#)
- `randpicks`: see [das\\_chop\\_equallength](#); NULL if using "condition" method

## 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 "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
das_effort(y.proc, method = "section", 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 <a href="#">das_effort</a> ; 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\_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**

<code>file</code>	character, the name of the file where the pdf will be saved
<code>...</code>	passed on to <code>file.copy</code> ; 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](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, y, ...)

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

**Arguments**

x	a data frame (such as an object of class <code>das_df</code> ) or a list. If x is a list, then it must be the output of <code>das_effort</code> or <code>das_effort_sight</code> . If x is a data frame, the user must also specify the coordinate columns of x using <code>x.lon</code> and <code>x.lat</code>
...	ignored
y	list of file path(s) of the CSV(s) that contain points defining each stratum. 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"

**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 `m lon` and `m lat`, 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.dttl.sight = FALSE`, because the ? and numeric event codes will have NA latitude and longitude values.

**Value**

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 `y`; the element(s) of `y` will have the name `InPoly#`, where '#' is the index of that stratum polygon in `y`.

If `x` is a list (meaning 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.dttl.sight = TRUE,
  ...
)

```

**Arguments**

x	either a <code>das_dfr</code> object, an object that can be coerced to class <code>das_dfr</code> , or a character (filepath) which is first passed to <code>das_read</code>
...	passed to <code>das_read</code> 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.dttl.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
- 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.
- 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.
- '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:

<i>State/condition</i>	<i>Column name</i>	<i>Data source</i>
On/off effort	OnEffort	B/R and E events
Cruise number	Cruise	Event: B; Column: Data1
Effort mode	Mode	Event: B; Column: Data2
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

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, tz = "UTC", ...)
```

### Arguments

file	filename(s) of one or more DAS files
skip	integer; see <a href="#">read_fwf</a> . Default is 0
tz	character; see <a href="#">strptime</a> . Default is "UTC"
...	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:

<i>Item</i>	<i>Columns</i>	<i>Format</i>
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 This function was inspired by [das.read](#)

### 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:

<i>Name</i>	<i>Class</i>	<i>Details</i>
EffortDot	logical	TRUE if "." was present, and FALSE otherwise
DateTime	POSIXct	combination of 'Date' and 'Time' columns with time zone tz
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 <code>file</code> argument
line_num	integer	line number of each data row

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_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 <code>das_df</code> , or a data frame that can be coerced to class <code>das_df</code>
...	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,

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.

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

<i>Sighting information</i>	<i>Column name</i>	<i>Notes</i>
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 otherwise
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-S/K/M/G events
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 $(\text{abs}(\sin(\text{Bearing} \times \pi / 180)) * \text{DistNm}) * 1.852$

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

One row for each species of each sighting event, with group size estimates calculated via an arithmetic mean across observer estimates (using `mean` with `na.rm = TRUE`)

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

<i>Sighting information</i>	<i>Column name</i>	<i>Notes</i>
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 of the product of GsSchoolBest and the corresponding species code
Group size of species - high estimate	GsSpHigh	Arithmetic mean of the product of GsSchoolHigh and the corresponding species code
Group size of species - low estimate	GsSpLow	Arithmetic mean of the product of GsSchoolLow and the corresponding species code

### 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:

<i>Sighting information</i>	<i>Column name</i>	<i>Notes</i>
Observer code - estimate	ObsEstimate	See below
Species 1 code	SpCode1	The species code for turtle and pinniped sightings
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
Sighting group size	TurtleGs	NA for non-"t" events; i.e. number of turtles
Was turtle captured?	TurtleCapt	NA for non-"t" events
Pinniped species	PinnipedSp	NA for non-"p" events
Sighting group size	PinnipedGs	NA for non-"p" events; i.e. number of pinnipeds
Boat or gear type	BoatType	NA for non-"F" events
Sighting group size	BoatGs	NA for non-"F" events; i.e. number of boats

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

### 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](https://en.wikipedia.org/wiki/Great-circle_distance)

---

randpicks_convert	<i>Convert randpicks file</i>
-------------------	-------------------------------

---

**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

**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**

<code>x</code>	object of class <code>das_dfr</code> or <code>das_df</code>
<code>i, j, ...</code>	elements to extract or replace, see <a href="#">[.data.frame]</a>
<code>drop</code>	logical, see <a href="#">[.data.frame]</a>
<code>name</code>	A literal character string or ..., see <a href="#">[.data.frame]</a>
<code>value</code>	A suitable replacement value, see <a href="#">[.data.frame]</a>

**Details**

When subsetting a `das_dfr` or `das_df` object, henceforth a `das_` object, using any of the functions described in [\[.data.frame\]](#), 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

subsetting a `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

- \* **package**
  - swfscDAS-package, 2
- .chop\_condition\_eff
  - (swfscAirDAS-internals), 28
- .chop\_equallength\_eff
  - (swfscAirDAS-internals), 28
- .dist\_from\_prev
  - (swfscAirDAS-internals), 28
- .process\_chr (swfscAirDAS-internals), 28
- .process\_num (swfscAirDAS-internals), 28
- .segdata\_aggr (swfscAirDAS-internals), 28
- .segdata\_proc (swfscAirDAS-internals), 28
- [.das\_df (subsetting), 27
- [.das\_dfr (subsetting), 27
- [.data.frame, 27
- [<-.das\_df (subsetting), 27
- [<-.das\_dfr (subsetting), 27
- [[<-.das\_df (subsetting), 27
- [[<-.das\_dfr (subsetting), 27
- \$<-.das\_df (subsetting), 27
- \$<-.das\_dfr (subsetting), 27
  
- as.numeric, 23
- as\_das\_df, 3, 8, 9, 28
- as\_das\_dfr, 3, 9, 10, 28
  
- das.read, 21
- das\_check, 4
- das\_chop\_condition, 11
- das\_chop\_equallength, 11, 12, 26
- das\_chop\_section, 12
- das\_comments, 6
- das\_df (das\_df-class), 8
- das\_df-class, 8
- das\_dfr (das\_dfr-class), 9
- das\_dfr-class, 9
- das\_effort, 10, 13, 15
- das\_effort\_sight, 12, 13, 15
  
- das\_format\_pdf, 14, 21–23
- das\_intersects\_strata, 15
- das\_process, 8, 9, 11, 16
- das\_read, 9, 16–18, 19
- das\_sight, 8, 12, 13, 22
- data.frame, 8, 9
- detectCores, 11
- distance, 12
- distance\_greatcircle, 12, 25
  
- file.copy, 14
  
- mean, 24
  
- randpicks\_convert, 26
- read\_fwf, 5, 19
  
- st\_intersects, 15
- strptime, 19
- subsetting, 27
- swfscAirDAS-internals, 28
- swfscDAS (swfscDAS-package), 2
- swfscDAS-package, 2
  
- table, 5
- toupper, 23