

Package: LTabundR (via r-universe)

October 4, 2026

Title Development Version

Version 1.75

Description Development version

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports cowplot, dplyr, DT, geosphere, ggbreak, ggplot2, ggpubr,
gridExtra, leaflet, lubridate, lutz, lwgeom, magrittr, mgcv,
miceadds, mrds, MuMIn, pkgcond, purrr, raster, rasterVis,
RColorBrewer, readr, readtext, remotes, rintrojs,
rnaturalearth, scales, scam, sf, shiny, shinydashboard,
shinyjs, shinythemes, sp, stats, stringr, swfscDAS, tibble,
tidyr, tools, units

Depends R (>= 2.10)

LazyData true

LazyDataCompression xz

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev git libglpk-dev make libicu-
dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libpoppler-cpp-dev poppler-
data libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

Repository <https://pifsc-protected-species-division.r-universe.dev>

Date/Publication 2026-09-24 14:02:04 UTC

RemoteUrl <https://github.com/PIFSC-Protected-Species-Division/LTabundR>

RemoteRef HEAD

RemoteSha 5f00b83c410959c79bedd4b946c303fadf1a62b2

Contents

abundance	4
barlow_2015	6

bca	7
cnp_150km_1986_2020	7
cnp_1986_2020_edits	10
coastline	11
coordinates_in_strata	11
cruz_combine	12
cruz_explorer	12
cruz_structure	13
das_editor	13
das_editor_tofile	14
das_format	15
das_inspector	16
das_interpolate	17
das_load	17
das_readtext	18
das_time	19
df_curve	19
df_fit	20
df_plot	22
eez	24
eez_ccs	24
eez_hawaii	25
er_simulator	25
example_cruz	26
example_settings	27
filter_cohort	28
filter_cruz	30
g0_combine	32
g0_model	32
g0_optimize	35
g0_plot	35
g0_results	36
g0_table	37
g0_weighted	38
gg_effort	40
gg_geostratum	41
gg_sightings	42
ggplot_idl	43
grp_calibrate_abund	44
grp_coeff_abund	45
grp_coeff_gerrodette	45
grp_ops_abund	46
grp_ops_gerrodette	46
grp_recalibrate	47
grp_size	48
idl_coerce	49
land	50
load_cohort_settings	50

load_settings	54
load_survey_settings	55
lta	58
lta_checks	66
lta_ci	67
lta_combine	68
lta_delta_method	69
lta_destratify	69
lta_diagnostics	70
lta_enlist	71
lta_estimates	72
lta_plot	73
lta_pool	74
lta_report	74
lta_result	75
lta_subgroup	77
map_cruz	82
mixed_group_affiliates	83
noaa_10km_1986_2020	84
points_to_line	85
prep_bootstrap_datasets	86
process_km	87
process_polygon	88
process_sightings	89
process_strata	90
process_subgroups	91
process_surveys	92
read_polygon	99
region_das	100
region_segments	100
segmentize	101
ships	103
species_codes	104
species_translator	104
strata_area	105
strata_ccs	107
strata_cnp	107
strata_etp	108
strata_explore	108
strata_segments	109
strata_select	110
subgroup_edits	110
subgroup_events	111
subgroup_explorer	112
subgroup_phases	112
subgroup_populations	113
subgroup_sightings	114
subgroup_subgroups	114

summarize_bft 115

summarize_distances 116

summarize_effort 116

summarize_sightings 117

summarize_species 118

Index 120

abundance	<i>Estimate density & abundance</i>
-----------	---

Description

This function is typically not called by the user (though it certainly can be); instead, it is called as a subroutine within `lta()`. This function estimates density/abundance for Wincruz data, as processed by `LTabundR::process_surveys()`.

Usage

```
abundance(  
  segments,  
  sightings,  
  das,  
  strata,  
  truncation_distance,  
  use_g0 = TRUE,  
  g0 = c(1, 1),  
  g0_threshold = 20,  
  region_pool = FALSE,  
  region_pool_name = "Regions pooled",  
  region_pool_area = NA,  
  year_pool = FALSE,  
  forced_effort = NULL,  
  verbose = TRUE  
)
```

Arguments

segments	Effort segments dataframe, drawn from a cruz object (e.g., <code>cruz\$cohorts[[1]]\$segments</code>), already filtered to contain the effort you wish to use to estimate density/abundance.
sightings	Sightings dataframe, drawn from a cruz object (e.g., <code>cruz\$cohorts[[1]]\$sightings</code>), already filtered to contain the sightings you wish to use to estimate the density/abundance. This sightings dataframe must have a column named <code>esw</code> , which can be provided by the <code>LTabundR</code> function <code>fit_df()</code> (fit a detection function model). If NA's occur in the <code>esw</code> column, they will be replaced with the mean <code>esw</code> value for the remainder of the dataset in that region-year. Similarly, if <code>sightings</code> has a column named <code>ss_valid</code> (all standard cruz objects do) and any of the rows in that column are <code>FALSE</code> , those rows will have their best school

	size estimate (which will be NA or 1, since they are invalid) replaced by the mean best estimate for their respective species.
<code>das</code>	Dataframe of DAS data, drawn from a <code>cruz</code> object (e.g., <code>cruz\$cohorts[[1]]\$das</code>)
<code>strata</code>	A dataframe, drawn from a <code>cruz</code> object (e.g., <code>cruz\$strata</code>), summarizing the geostrata provided (their name and area, in square km).
<code>truncation_distance</code>	The truncation distance to apply to sightings.
<code>use_g0</code>	A Boolean, with default TRUE, indicating whether or not to use custom $g(0)$ value(s). If FALSE, the assumed $g(0)$ value will be 1.
<code>g0</code>	A numeric vector of length 2: the $g(0)$ for small and large groups.
<code>g0_threshold</code>	The school size threshold between small and large groups.
<code>region_pool</code>	Accepts a Boolean; if TRUE, the functions will produce a single estimate for all geostrata combined; if FALSE, the function calculates density and abundance for each geostratum contained within the data (according to <code>segments\$stratum</code>) separately. If you set <code>region_pool</code> to TRUE, the next two arguments ought to be specified.
<code>region_pool_name</code>	A character string indicating a custom name to use in the Region column in the output. Only used in the event that <code>region_pool == TRUE</code> .
<code>region_pool_area</code>	A numeric indicating the area, in square km, of the pooled region. Only used in the event that <code>region_pool == TRUE</code> . If left NULL but <code>region_pool == TRUE</code> , abundance will not be calculated.
<code>year_pool</code>	A Boolean indicating whether or not to pool years (set <code>year_pool</code> to TRUE) or to report density/abundance for each year separately (set to FALSE).
<code>forced_effort</code>	If this is a single numeric value instead of NULL (NULL is the default), this value will be used as the survey effort, in km, in a brute-force method; this same value will be used for every year and region. This is only helpful if you are looking for a relatively easy way to compare results from your own analysis to another (e.g., comparing LTabundR results to reports from NOAA reports prior to 2021, in which effort was calculated slightly differently).
<code>verbose</code>	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Value

A `data.frame` with a row for each density/abundance estimate. You may expect multiple rows if multiple regions or years are contained within the datasets provided. This `data.frame` contains the following fields:

1. `Region`: Name(s) of geostrata represented in this estimate.
2. `Area`: Area of geostratum / region, in square km.
3. `year`: Years represented in this estimate.
4. `segments`: The number of effort segments used to estimate density/abundance.
5. `km`: The km of trackline effort contained in these segments.

6. Area_covered: The Area surveyed, according to km and ESW_mean (see next column).
7. ESW_mean: Mean effective strip width, in kw, calculated as the truncation distance multiplied by the mean probability of detection for all detections.
8. n: The number of detections in the data.
9. g0_mean: The mean $g(0)$ estimate.
10. ER_clusters: The encounter rate for detections (schools) (n / km)
11. D_clusters: The density of detections (schools).
12. N_clusters: The abundance of schools.
13. size_mean: Average school size.
14. size_sd: Standard deviation of school size.
15. ER: Animal encounter rate.
16. D: Animal density.
17. N: Animal abundance.

barlow_2015

Relative trackline detection probabilities from Barlow (2015).

Description

Relative trackline detection probabilities from Barlow (2015).

Usage

barlow_2015

Format

An object of class `data.frame` with 168 rows and 9 columns.

Details

This dataset holds the results from Barlow (2015), "Inferring trackline detection probabilities, $g(0)$, for cetaceans from apparent densities in different survey conditions" (*Marine Mammal Science*), which used NOAA/NMFS cruise data from 1986 to 2010.

bca	<i>BCA method function</i>
-----	----------------------------

Description

Replicated from the coxed package, which has been archived on CRAN. Moving code here so that LTabundR can still use this function.

Usage

```
bca(theta, conf.level = 0.95)
```

Arguments

theta	Expectation.
conf.level	Confidence level.

Value

A two-element vector with the lower and upper confidence interval calculated using the BCA method.

cnp_150km_1986_2020	<i>Cruise data processed for the CNP, 1986 - 2020 (150 km segment lengths)</i>
---------------------	--

Description

Cruise data processed for the CNP, 1986 - 2020 (150 km segment lengths)

Usage

```
cnp_150km_1986_2020
```

Format

An object of class `list` of length 3.

Details

This dataset was processed with the following code:

```
data(group_size_coefficients)

survey <- load_survey_settings(
  out_handling = 'remove',
  max_row_interval = Inf,
  segment_method = "equallength",
  segment_target_km = 150,
  segment_max_interval = 24,
  segment_remainder_handling = c("segment"),
  ship_list = NULL, # use package list
  species_codes = NULL, # use package codes
  group_size_coefficients = group_size_coefficients, # use package coefficients
  smear_angles = FALSE)

data(strata_cnp)
strata <- strata_cnp

all_species <- load_cohort_settings(
  id = "all",
  species = NULL, # spp_codes,
  strata = c('MHI', 'WHICEAS', 'HI_EEZ', 'OtherCNP'),
  probable_species = FALSE,
  sighting_method = 0,
  cue_range = 0:7,
  school_size_range = c(0, 10000),
  school_size_calibrate = TRUE,
  calibration_floor = 0,
  use_low_if_na = TRUE,
  io_sightings = 0,
  geometric_mean_group = TRUE,
  truncation_km = 7.5,
  beaufort_range = 0:6,
  abeam_sightings = FALSE,
  strata_overlap_handling = c("smallest"),
  distance_types = c('S', 'F', 'N'),
  distance_modes = c('P', 'C'),
  distance_on_off = TRUE
)

bottlenose <- load_cohort_settings(
  id = "bottlenose",
  species = '018',
  strata = c('MHI', 'WHICEAS', 'HI_EEZ', 'OtherCNP', 'Bottlenose_BI', 'Bottlenose_OUFI', 'Bottlenose_Ka'),
  probable_species = FALSE,
  sighting_method = 0,
```

```

cue_range = 0:7,
school_size_range = c(0, 10000),
school_size_calibrate = TRUE,
calibration_floor = 0,
use_low_if_na = TRUE,
io_sightings = 0,
geometric_mean_group = TRUE,
truncation_km = 7.5,
beaufort_range = 0:6,
abeam_sightings = FALSE,
strata_overlap_handling = c("smallest"),
distance_types = c('S','F','N'),
distance_modes = c('P','C'),
distance_on_off = TRUE
)

spotted <- load_cohort_settings(
id = "spotted",
species = '002',
strata = c('MHI', 'WHICEAS', 'HI_EEZ', 'OtherCNP', 'Spotted_OU', 'Spotted_FI', 'Spotted_BI'),
probable_species = FALSE,
sighting_method = 0,
cue_range = 0:7,
school_size_range = c(0, 10000),
school_size_calibrate = TRUE,
calibration_floor = 0,
use_low_if_na = TRUE,
io_sightings = 0,
geometric_mean_group = TRUE,
truncation_km = 7.5,
beaufort_range = 0:6,
abeam_sightings = FALSE,
strata_overlap_handling = c("smallest"),
distance_types = c('S','F','N'),
distance_modes = c('P','C'),
distance_on_off = TRUE
)

fkw <- load_cohort_settings(
id = "pseudorca",
species = '033',
strata = c('MHI', 'WHICEAS', 'HI_EEZ', 'OtherCNP', 'NHWI'),
probable_species = FALSE,
sighting_method = 0,
cue_range = 0:7,
school_size_range = c(0, 10000),
school_size_calibrate = TRUE,
calibration_floor = 0,

```

```

use_low_if_na = TRUE,
io_sightings = 0,
geometric_mean_group = TRUE,
truncation_km = 7.5,
beaufort_range = 0:6,
abeam_sightings = FALSE,
strata_overlap_handling = c("smallest"),
distance_types = c('S','F','N'),
distance_modes = c('P','C'),
distance_on_off = TRUE
)

settings <- load_settings(strata = strata,
                        survey = survey,
                        cohorts = list(all_species,
                                      bottlenose,
                                      spotted,
                                      fkw))

das_file = '../test_code/CNP/CenPac1986-2020_Final_alb.das'

cnp_150km_1986_2020 <- process_surveys(das_file = das_file,
                                       settings = settings)

```

cnp_1986_2020_edits *Example of coded edits to cruz object*

Description

Example of coded edits to cruz object

Usage

```
cnp_1986_2020_edits
```

Format

An object of class `list` of length 4.

Details

This is an example of the kind of object that can be passed to the `edits` input in `process_surveys()`.

coastline	<i>Coastline</i>
-----------	------------------

Description

Coastline

Usage

coastline

Format

An object of class sf (inherits from data.frame) with 4133 rows and 4 columns.

Details

World coastline, downloaded from Natural Earth.

coordinates_in_strata	<i>Determine whether coordinates occur inside a stratum polygon</i>
-----------------------	---

Description

This is an internal function typically not called by a user directly.

Usage

coordinates_in_strata(lon, lat, poli)

Arguments

- | | |
|------|---|
| lon | Longitude coordinates in decimal degrees, in which West coordinates are negative. |
| lat | Latitude coordinates in decimal degrees, in which South coordinates are negative. Must be the same length as lon. |
| poli | A dataframe of polygon coordinates. |

Value

A logical vector, the same length as the lon, indicating whether or not the coordinate falls within the polygon. If the coordinate falls exactly on the polygon boundary, it will be counted as in.

cruz_combine	<i>Combine several cruz objects</i>
--------------	-------------------------------------

Description

This function combines processed cruz objects (the outputs of `LTabundR::process_surveys()`), under the assumption that the survey settings in both objects are the exact same. If the same cohort name occurs in multiple cruz objects, the contents of the instances of the cohort are checked for redundancy (using Cruise number - date combinations), and only non-redundant content is combined. If different cohorts occur in the supplied cruz objects, the cohorts are added without modification.

Usage

```
cruz_combine(cruzes)
```

Arguments

cruzes	A list of cruz objects, e.g., <code>list(cruz1, cruz2, cruz3)</code> . To understand cruz objects, see the output of the function <code>process_surveys()</code> .
--------	--

Value

A single cruz object.

cruz_explorer	<i>Launch a data explorer app for WinCruz surveys</i>
---------------	---

Description

Launch a data explorer app for WinCruz surveys

Usage

```
cruz_explorer(cruz, cohort = 1)
```

Arguments

cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).
cohort	The cohort whose data you would like to map, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.

Value

A Shiny app for interactive data exploration. This app allows you to filter the cruz object according to various fields, then explore summary tables of effort, sightings, and detection distances. There are also tabs for reviewing the data in tabular form and exploring an interactive leaflet map of survey tracks, geostrata, and sightings.

cruz_structure	<i>Review of cruz object structure</i>
----------------	--

Description

This function prints an overview of the list structure and sample sizes within your cruz object, which is produced from `LTabundR::process_surveys()`.

Usage

```
cruz_structure(cruz)
```

Arguments

`cruz` Your cruz object (produced from `LTabundR::process_surveys()`).

Value

Nothing; messages are printed to the Console.

das_editor	<i>Apply edits to a DAS file...</i>
------------	-------------------------------------

Description

...without modifying the original data. Edits are provided as a list, which the user can store as a `RData` object or simply prepare with an R script. This function loops through each edit and applies them to the respective DAS file. This allows survey data to be modified reproducibly before being processed with `LTabundR::process_surveys()`, without touching the original DAS data files or requiring analysts to duplicate files and make one-off modifications manually.

Usage

```
das_editor(edits)
```

Arguments

`edits` A list of sublists, each being a staged edit. Each sublist requires the following named slots:

- `das_file`: Filepath to a DAS file.
- `type`: The type of edit. Currently accepted options are:
 - `"text"`: the edit will be interpreted verbatim as text that will replace the specified data. Both rows and characters must be supplied, as well as the edit itself.

- "function": the edit will be evaluated as a function that is applied to each row of specified data. Both rows and characters must be supplied, as well as edit. For example, "tolower" will convert the specified data to all lower case. Another example: "function(x){LTabundR::das_time(x, -10)}" will subtract 10 hours from the data if your specified chars is the date-timestamp for a row.
- "move": the rows will be deleted from their current location and pasted immediately below the row number specified by edit. The moved rows will be given the same date, time, latitude, and longitude, as the edit row. Only rows and edits need to be supplied.
- "copy": the rows will be copied from their current location and pasted immediately below the row number specified by edit. The paste rows will be given the same date, time, latitude, and longitude, as the edit row. Only rows and edits need to be supplied.
- "insert": the text provided in edit will be inserted verbatim immediately below the first of the rows provided. Only rows and edits needs to be supplied.
- "delete": the rows will be deleted. Only rows needs to be supplied.
- rows: A numeric vector of the rows to be edited.
- chars: The character indices to be edited; e.g., "1:3" specifies that only the first three characters in each of rows will be edited.
- edit: The actual edit, provided as a vector whose format depends on the type chosen above. See examples.

This edits list of lists can be as long as necessary, and it can contain as edits for as many DAS files as needed. The das table that is returned by the function will concatenate all das files represented by the edits into a single table.

Value

A list with two named slots:

- das: A data.frame containing the modified DAS data (with one column: \$das).
- log: The edits input, with each sublist augmented with a results slot that shows the results of the edit.

das_editor_tofile

Edit DAS files and save edited versions

Description

This function is basically a wrapper for the function `das_editor()`, but this function saves the edited DAS data to new files. It takes a set of DAS files and a set of editing instructions (provided as a list – see `das_editor()` documentation), applies the edits to each DAS file, and saves new versions of the DAS file. It does *NOT* replace the original DAS file.

Usage

```
das_editor_tofile(das_file, edits, suffix = "_edited", verbose = TRUE)
```

Arguments

das_file	A vector of DAS data filename(s).
edits	A list of sublists, each being instructions for a staged edit. See ?das_editor() for details.
suffix	The suffix to append to DAS files to distinguish the original file from the new edited file.
verbose	Boolean. Printed updates to the Console?

Value

At the end of the process, the function returns a vector of DAS filenames that should be passed to processing functions such as process_surveys() or das_load().

das_format	<i>Finalize formatting of a DAS object</i>
------------	--

Description

This function removes invalid rows of data (due to missing Cruise numbers, timestamps, location coordinates, etc.), calculates the distance, in kilometers, between each row of data, adds a ship column with the ship name associated with the cruise number, and initiates the cruz object structure, with a new cohorts slot.

This is an internal function typically not called by a user directly. It is the third subroutine called within process_surveys(); after process_strata() and before segmentize().

Usage

```
das_format(cruz, verbose = FALSE)
```

Arguments

cruz	The list produced by process_strata(), with two slots: settings and das.
verbose	Boolean, with default FALSE, indicating whether or not updates should be printed to the Console.

Value

A list with nascent cruz object structure, with these slots:

1. settings holds your settings object;
2. strata holds a dataframe summarizing the name and area (square km) of each geo-stratum, if any are provided;
3. cohorts holds a list with slots for each cohort of species as specified in settings. Each cohort slot has a copy of the DAS data with a new stratum column, with a stratum assignment tailored to its cohort-specific settings (specifically, the setting stratum_overlap_handling). This list structure will be expanded upon in subsequent steps of process_surveys().

das_inspector

Use a Shiny app to explore a DAS file and build up a list of edits. Those edits will be saved as a list of instructions, which can subsequently be provided as an argument to LTabundR::process_surveys(), which will apply your list of edits before processing the data. This framework allows you to record details of required edits reproducibly without ever modifying the original dataset or producing modified versions of the original.

Description

Use a Shiny app to explore a DAS file and build up a list of edits. Those edits will be saved as a list of instructions, which can subsequently be provided as an argument to LTabundR::process_surveys(), which will apply your list of edits before processing the data. This framework allows you to record details of required edits reproducibly without ever modifying the original dataset or producing modified versions of the original.

Usage

```
das_inspector(das_file)
```

Arguments

das_file Path to your DAS file.

Value

A Shiny app is launched, where the user explores and builds up a list of staged edits. Once the user closes that app, this function returns that list of edits. We recommend saving that return in an object for easy call back later. To understand the structure of the list of edits, see LTabundR::das_editor().

das_interpolate	<i>Interpolate DAS data</i>
-----------------	-----------------------------

Description

This function is not typically called by the user directly; it is called during `process_surveys()`. The user can instruct LTabundR to interpolate DAS data in `load_survey_settings()`.

Usage

```
das_interpolate(das, new_interval = 120, max_ignore = 3600, verbose = FALSE)
```

Arguments

<code>das</code>	A <code>data.frame</code> of a DAS survey data file, created by <code>das_load()</code> .
<code>new_interval</code>	The interpolation interval, in seconds.
<code>max_ignore</code>	The maximum interval (seconds) between two rows of survey data that will be interpolated; if the time gap exceeds this value, interpolation will not occur for these two rows. The default is 3600 seconds (1 hour).
<code>verbose</code>	Print updates to the Console?

Details

This function allows you to interpolate the DAS position data at the onset of processing if your position updates are separated by large time intervals, which would make spatial effort and stratum assignments less exact. LTabundR will interpolate the data using simple-linear methods (i.e., no great-circle calculations), such that position updates occur every `new_interval` seconds or less. If adjacent DAS rows are from different dates or cruises, or if the interval between rows exceeds the input `max_ignore`, the interpolation routine will skip to the next pair of related rows. Interpolation will only occur for On-Effort rows.

Value

An interpolated `data.frame` of the DAS data. No formatting has been changed.

das_load	<i>Read in and process a DAS file</i>
----------	---------------------------------------

Description

This is an internal function typically not called by a user directly. It is the first subroutine called within `process_surveys()`. This function is essentially a wrapper for `swfscDAS::das_read()`, which reads in a raw DAS file (or a URL to an online repository), and `swfscDAS::das_process()`, which formats each row into columns, with survey status, sighting conditions, and observer positions added to each row as new columns.

Usage

```
das_load(das_file, perform_checks = FALSE, print_glimpse = FALSE)
```

Arguments

<code>das_file</code>	Filepath to WinCruz DAS file with survey data. Filepath can be absolute or relative. It can also be a URL.
<code>perform_checks</code>	Boolean, with default of FALSE; should <code>swfscDAS::das_check()</code> be run first, with diagnostics printed to the console?
<code>print_glimpse</code>	Boolean, with default of FALSE; print a glimpse (<code>dplyr::glimpse()</code>) of the formatted DAS dataframe?

Value

A `data.frame` of the DAS file, with fields parsed into columns and with new columns for survey status, sighting conditions, and observer positions.

<code>das_readtext</code>	<i>Read DAS file without parsing columns</i>
---------------------------	--

Description

In contrast to `swfscDAS::das_read()`, this function reads the DAS text in, saving each line of data to a single row of a one-column `data.frame` (no column parsing). This is useful if you wish to edit and save an altered version of the data using `LTabundR::das_inspector()` and `LTabundR::das_editor()`. This function relies upon the `readtext` package.

Usage

```
das_readtext(das_file)
```

Arguments

<code>das_file</code>	Filepath to DAS file.
-----------------------	-----------------------

Value

A one-column `data.frame` (column name is `das`).

das_time	<i>Adjust the timestamp within a DAS file</i>
----------	---

Description

This function is typically called within LTabundR: :das_inspector() as a way of staging an edit that adjusts the timestamps of certain rows. Not typically called directly by a user.

Usage

```
das_time(das, tz_adjust = "from utc")
```

Arguments

das	A row of DAS data. It may already be cropped to the characters corresponding to timestamp, date, lat, and long (characters 6 - 40).
tz_adjust	One of two options: (1) a single numeric value indicating the hours to add to or subtract from the date-time in each DAS row, or (2) the character string "from utc", which will indicate that the DAS timestamps are in UTC and need to be converted to local time. To do this, the latitude and longitude of each row will be used to determine the timezone of the event, then adjust the timestamp from UTC accordingly. This process can take several minutes.

Value

A two-slot list: [[1]] dt returns the revised date-time characters. [[2]] das returns the revised row in full (if a full row was originally provided).

df_curve	<i>Get detection function curve</i>
----------	-------------------------------------

Description

This is an internal function, typically not called by the user. It returns a single detection function curve based on one or more detection function models. If multiple models are provided, their curves will be averaged together to provide a single curve, weighting the average based upon the model AIC.

Usage

```
df_curve(models, covariates = NULL, truncation_distance)
```

Arguments

models	A list of best-fitting detection function model(s) (produced by <code>LTabundR::fit_df()</code>).
covariates	If NULL, the function will assume that this detection function does not use covariates. If covariates <i>are</i> used, this input should be anything <i>except</i> NULL.
truncation_distance	Truncation distance used, in km.

Details

This function is adapted from code in `mrds::plot.ds()` and `mrds::detfct()`.

Value

A `data.frame` of length 101, with two columns: `km` is distance from the trackline, and `p` is the average detection probability at that distance, based on the data's covariates.

df_fit	<i>Fit a detection function to WinCruz data</i>
--------	---

Description

This function is not typically called by the user (though it can be); instead it is called as a subroutine from within the function `lta()`. This function is a wrapper for the detection function fitting routine in the package `mrds` (function `mrds::ddf()`). As a wrapper, this function conducts automated stepwise model-fitting, trying all candidate function keys and covariates, then selecting the best model(s) based upon AICc.

Usage

```
df_fit(
  sightings,
  truncation_distance = Inf,
  covariates = NULL,
  detection_function_base = "hn",
  base_model = "~1",
  delta_aic = 2,
  topplot = TRUE,
  verbose = TRUE
)
```

Arguments

sightings	Sightings dataframe, drawn from a <code>cruz</code> object (e.g., <code>cruz\$cohorts[[1]]\$sightings</code>), already filtered to contain the sightings you wish to use to fit the detection function.
truncation_distance	The truncation distance to apply to sightings before model fitting.

covariates	Covariates you wish to include as candidates in detection function models, provided as a character vector. The covariates must match columns existing within sightings. Common covariates that you will find within sightings include <code>c('Bft', 'LnSsTot', 'Cruise', 'Year', 'Ship', 'species')</code> . Note that the function will ignore case, coercing all covariates to lowercase.
detection_function_base	The base key for the detection function, provided as a character vector. Accepted values are "hn" (half-normal key, the default, which exhibit greater stability when fitting to cetacean survey data; Gerrogette and Forcada 2005), "hr" (hazard-rate), or <code>c("hn", "hr")</code> , which will loop through both keys and attempt model fitting.
base_model	The initial model formula, upon which to build using candidate covariates, provided as a character vector. The default is " <code>~ 1</code> ".
delta_aic	The AIC difference between the model yielding the lowest AICc and other candidate models, used to define the best-fitting models. Typically, AICc differences of less than 2 indicate effectively equal model performance.
topplot	Boolean, with default TRUE, indicating whether detection function plots (<code>Distance::plot.ds()</code>) should be displayed as the candidate models are tested.
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Details

Model fitting is done in a forward stepwise procedure, starting fresh with each base key provided. In the first round, the base model (no covariates) is fit first. In the second round, each covariate is added one at a time; the covariate, if any, that produces the lowest AICc below the AIC from the previous round is added to the formula. This process is repeated in subsequent rounds until the AICc no longer improves. All models within `delta_aic` of the model with the lowest AICc qualify as best-fitting models.

Value

A named list:

1. `best_models`: A `data.frame` summary of the best-fitting models, based upon the table produced by `Distance::summarize_ds_models()`. See that function's documentation for details.
2. `all_models`: Similar to the preceding slot, a tabular summary of *all* models tested.
3. `best_objects`: A list containing the `ds` objects (produced by package `Distance`) for each of the best-fitting models.
4. `sample_size`: A `data.frame` with the detections for each species within the species pool used to fit the detection function. `Ntot` is total detections for each species; `Ndet` is total detections within the truncation distance and therefore used in the detection function fitting routine; `TD` is the truncation distance.
5. `tables`: A list of the data tables passed to `Distance::ds()` during model fitting.
6. `sightings`: The sightings `data.frame` provided, now with an `esw` column provided estimated Effective Strip half-Width for each sighting.

df_plot

*Plot the best-fit detection function model(s)***Description**

This function adapts `mrds::plot.ds()` to plot the detection functions fit and provided by the output of `lta()` and `lta_multistock()`. Arguments allow for stylizing the plot as well as displaying multiple best-fitting models on the same histogram.

Usage

```
df_plot(
  lta_result,
  model_colors = "black",
  model_pch = 1,
  histBars = 15,
  hist_col = "grey90",
  hist_border = "grey60",
  pt_show = 1,
  pt_cex = 0.8,
  pt_alpha = 0.3,
  line_col = "darkblue",
  line_lty = 1,
  line_lwd = 2.5,
  line_alpha = 0.7,
  bootstrap_show = TRUE,
  bootstrap_col = "steelblue4",
  bootstrap_alpha = 0.2,
  bootstrap_lwd = 0.5,
  main_show = TRUE,
  main = NULL,
  main_cex = 1.25,
  legend_show = FALSE,
  ymax = NULL,
  legend_x = 3.2,
  legend_y = 1.2,
  legend_cex = 0.8,
  xlab = "Distance (km) from trackline"
)
```

Arguments

<code>lta_result</code>	The result of <code>lta()</code> for a single species pool.
<code>model_colors</code>	A vector of colors; if more than one is provided, the vector must be the same length as the number of best-fitting models (<code>lta_result\$df\$best_models</code>)
<code>model_pch</code>	A vector of point pch, of either length 1 or the same as the number of best-fitting models.

histBars	Number of histogram bars to show.
histCol	Color of histogram bars.
histBorder	Border color of histogram bars.
ptShow	If 0, no detection probabilities for individual detections will be shown. If 1, only detection probabilities for the first best-fitting model will be shown, as points. (This is the default.) If 2, points will be displayed for <i>all</i> best-fitting models.
ptCex	The size of detection points, if shown.
ptAlpha	The transparency (0 = transparent; 1=solid) of points, if shown.
lineCol	Color of detection function line.
lineLty	The line type for the detection function, following conventional base plot lty values.
lineLwd	Detection function line thickness.
lineAlpha	Detection function line transparency, as a fraction between 0 and 1.
bootstrapShow	Show detection function curves for all bootstrap iterations? Default is TRUE.
bootstrapCol	Color of bootstrap detection function curves, if shown.
bootstrapAlpha	Transparency of bootstrap detection function curves, if shown.
bootstrapLwd	Thickness of bootstrap detection function curves, if shown.
mainShow	Show a main title? TRUE (default) or FALSE.
main	The text for the main title; if NULL, the species pool name will be provided.
mainCex	Size of main title.
legendShow	Show a legend, indicating the line/point/color for each best-fitting model? TRUE or FALSE (default).
ymax	Maximum of y-axis (can be helpful when making space for the legend). Can be NULL.
legendX	Left-right position of legend, if shown.
legendY	Up-down position of legend, if shown.
legendCex	Size of legend text, if shown.
xlab	Text for x axis label, if shown.

Value

A plot is printed.

eez	<i>EEZ (all US EEZ boundaries)</i>
-----	------------------------------------

Description

EEZ (all US EEZ boundaries)

Usage

eez

Format

An object of class sf (inherits from data.frame) with 260 rows and 17 columns.

Details

World dataset of geopolitical Exclusive Economic Zones (EEZ), downloaded from Natural Earth.

eez_ccs	<i>EEZ - California Current System (formatted for sf mapping)</i>
---------	---

Description

EEZ - California Current System (formatted for sf mapping)

Usage

eez_ccs

Format

An object of class sfc_MULTILINESTRING (inherits from sfc) of length 17.

Details

The EEZ relevant to the California Current System only, formatted for sf compatibility.

eez_hawaii	<i>EEZ - Hawaii (formatted for sf mapping)</i>
------------	--

Description

EEZ - Hawaii (formatted for sf mapping)

Usage

```
eez_hawaii
```

Format

An object of class `sfc_MULTILINESTRING` (inherits from `sfc`) of length 25.

Details

The EEZ relevant to the Hawai'ian area only, formatted for sf compatibility.

er_simulator	<i>Encounter Rate simulator test</i>
--------------	--------------------------------------

Description

Test for the probability that year-to-year changes observed in a species' encounter rate are due to random sampling variation instead of an actual change in the encounter rate. This function uses bootstrap sampling of survey segments to see if random variation in sampling could possibly produce an *apparent* but immaterial change in encounter rate across years.

Usage

```
er_simulator(  
  spp,  
  cohort = 1,  
  cruz,  
  iterations = 1000,  
  seed = NULL,  
  verbose = FALSE  
)
```

Arguments

spp	Species code
cohort	The cohort whose data you would like to analyze, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.
cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).
iterations	Number of iterations
seed	Set a seed (any integer) to ensure that the result is reproducible. If left NULL, the results are liable to differ for each run of this function.
verbose	Boolean; print updates to the console?

Details

See the Appendix to Bradford et al. (2020) for analytical details, but briefly: in each bootstrap iteration, survey segments are resampled in a way that preserves the proportion of effort occurring within each geostratum in the data. The resampled data are used to calculate the overall encounter rate across all years, since the null hypothesis is that the encounter rate does not change across years. This overall encounter rate is used to predict the number of sightings in each year, based on the distance covered by the resampled segments in each year. This process is repeated (`iterations` times) to produce a distribution of predicted sighting counts in each year. This distribution is compared to the actual number of sightings observed in each year. The number of simulated sightings counts that exceed the observed count reflects the probability that the observed count is due to random sample variation alone.

Value

A dataframe with a row for each year. Columns provide the number of observations of the species of interest during systematic effort, and the p-value of the test. The p-value represents the fraction of simulated encounter rates that exceed the observed encounter rate.

example_cruz	<i>Example cruz object (WHICEAS 2020)</i>
--------------	---

Description

Example cruz object (WHICEAS 2020)

Usage

```
example_cruz
```

Format

An object of class `list` of length 3.

filter_cohort	<i>Filter segments & sightings for a single cohort by species/region/year, etc.</i>
---------------	---

Description

This is an internal function, usually not called directly by analysts, but it certainly can be. Rather than calling a cruz object, as `filter_cruz` does, this function just calls for the three relevant constituent tables: segments, sightings, and the full das dataset.

Usage

```
filter_cohort(
  segments,
  sightings,
  das,
  analysis_only = FALSE,
  spp = NULL,
  years = NULL,
  not_years = NULL,
  cruises = NULL,
  not_cruises = NULL,
  regions = NULL,
  not_regions = NULL,
  bft_range = NULL,
  eff_types = NULL,
  on_off = NULL,
  lat_range = NULL,
  lon_range = NULL,
  verbose = TRUE
)
```

Arguments

segments	Effort segments dataframe, drawn from a cruz object (e.g., <code>cruz\$cohorts[[1]]\$segments</code>)
sightings	Sightings dataframe, drawn from a cruz object (e.g., <code>cruz\$cohorts[[1]]\$sightings</code>)
das	Dataframe of DAS data, drawn from a cruz object (e.g., <code>cruz\$cohorts[[1]]\$das</code>)
analysis_only	If TRUE, data will be filtered to those viable for use in formal analysis. For segments, this means rows in which <code>segments\$use == TRUE</code> . For sightings, this means rows in which <code>sightings\$included == TRUE</code> .
spp	A character vector of species codes, used to filter sightings based on <code>sightings\$species</code> .
years	A numeric vector of years, used to filter both segments and sightings to include only data from these years.
not_years	A numeric vector of years, used to filter both segments and sightings to include only data <i>not</i> from these years. Usually not useful and can be left as NULL,

	but may be useful if your data contain many years and it can be more efficient to exclude certain years rather than specify most years in the preceding argument.
cruises	A numeric vector of cruise numbers, used to filter a la years above.
not_cruises	A numeric vector of cruise numbers, used to filter a la not_years above.
regions	A character vector of geostratum names, used to filter both segments and sightings. Any segment or sighting occurring within <i>any</i> (<i>not</i> all) of the provided regions will be returned. This holds for nested regions; for example, in analyses from the Central North Pacific, in which the Hawaii EEZ geostratum ("HI-EEZ") is nested within the larger geostratum representing the entire CNP study area ("OtherCNP"), an input of regions = "OtherCNP" will return segments/sightings <i>both</i> inside the Hawaii EEZ <i>and</i> outside of it.
not_regions	A character vector of geostratum names, similar to above. Any segment or sighting occurring within any of these not_regions will not be returned. Using the example above, if regions = "OtherCNP" and not_regions = "HI-EEZ", only segments occurring within OtherCNP <i>and</i> outside of HI-EEZ will be returned. This can be particularly useful for abundance estimates for pelagic stock that exclude nested insular stocks.
bft_range	The Beaufort Sea State values to filter to, provided as a numeric vector.
eff_types	The effort types ('S' = Systematic, 'F' = Fine-scale, 'N' = Non-systematic) to filter to, provided as a character vector.
on_off	The OnEffort values to filter to (TRUE and/or FALSE), provided as a logical vector.
lat_range	If not NULL, a two-element numeric vector indicating latitudinal range of acceptable data. Accepted values are -90 (S) to 90 (N).
lon_range	If not NULL, a two-element numeric vector indicating latitudinal range of acceptable data. Accepted values are -180 (W) to 180 (E).
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Details

This function works by using the filter arguments to determine which rows of DAS data meet all criteria; since that DAS dataframe is drawn from a processed cruz object, it has a seg_id column indicating the segment IDs pertaining to eligible rows of data. Those segment IDs are then used to filter segments and sightings.

Note that when arguments are NULL, they are ignored and no filtering by the associated variable occurs.

Value

A list with two objects: segments and sightings. These are filtered versions of the same objects contained within the cruz object. See process_surveys() documentation for details.

filter_cruz

*Filter cruz objects by species/region/year, etc.***Description**

Subsets a cruz object according to a variety of filters. The same filters apply to all cohorts within the cruz object.

Usage

```
filter_cruz(
  cruz,
  analysis_only = FALSE,
  spp = NULL,
  years = NULL,
  not_years = NULL,
  cruises = NULL,
  not_cruises = NULL,
  regions = NULL,
  not_regions = NULL,
  bft_range = NULL,
  eff_types = NULL,
  on_off = NULL,
  lat_range = NULL,
  lon_range = NULL,
  verbose = TRUE
)
```

Arguments

cruz	Your cruz object (produced from LTabundR::process_surveys()).
analysis_only	If TRUE, data will be filtered to those viable for use in formal analysis. For segments, this means rows in which segments\$use == TRUE. For sightings, this means rows in which sightings\$included == TRUE.
spp	A character vector of species codes, used to filter sightings based on sightings\$species.
years	A numeric vector of years, used to filter both segments and sightings to include only data from these years.
not_years	A numeric vector of years, used to filter both segments and sightings to include only data <i>not</i> from these years. Usually not useful and can be left as NULL, but may be useful if your data contain many years and it can be more efficient to exclude certain years rather than specify most years in the preceding argument.
cruises	A numeric vector of cruise numbers, used to filter a la years above.
not_cruises	A numeric vector of cruise numbers, used to filter a la not_years above.

regions	A character vector of geostratum names, used to filter both segments and sightings. Any segment or sighting occurring within <i>any</i> (not all) of the provided regions will be returned. This holds for nested regions; for example, in analyses from the Central North Pacific, in which the Hawaii EEZ geostratum ("HI-EEZ") is nested within the larger geostratum representing the entire CNP study area ("OtherCNP"), an input of regions = "OtherCNP" will return segments/sightings <i>both</i> inside the Hawaii EEZ <i>and</i> outside of it.
not_regions	A character vector of geostratum names, similar to above. Any segment or sighting occurring within any of these not_regions will not be returned. Using the example above, if regions = "OtherCNP" and not_regions = "HI-EEZ", only segments occurring within OtherCNP <i>and</i> outside of HI-EEZ will be returned. This can be particularly useful for abundance estimates for pelagic stock that exclude nested insular stocks.
bft_range	The Beaufort Sea State values to filter to, provided as a numeric vector.
eff_types	The effort types ('S' = Systematic, 'F' = Fine-scale, 'N' = Non-systematic) to filter to, provided as a character vector.
on_off	The OnEffort values to filter to (TRUE and/or FALSE), provided as a logical vector.
lat_range	If not NULL, a two-element numeric vector indicating latitudinal range of acceptable data. Accepted values are -90 (S) to 90 (N).
lon_range	If not NULL, a two-element numeric vector indicating latitudinal range of acceptable data. Accepted values are -180 (W) to 180 (E).
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Details

This function works by using the filter arguments to determine which rows of DAS data meet all criteria; since that DAS dataframe comes from within the processed cruz object, it has a seg_id column indicating the segment IDs pertaining to eligible rows of data. Those segment IDs are then used to filter segments, sightings, and subgroups.

Note that the analysis_only filter is **ignored** for subgroups sightings/subgroups/events, since those are often case-by-case-specific.

Note that when arguments are NULL, they are ignored and no filtering by the associated variable occurs. Also note that subgroups are not filtered at this time.

Note that there is a similar function, filter_cohort(), that performs a similar task for only a single cohort, without requiring an in-tact cruz object. Instead it asks for the requisite datasets separately.

Value

A modified cruz object. See process_surveys() documentation for details.

g0_combine	<i>Combine estimates of weighted g(0) and its CV</i>
------------	--

Description

This function is typically not called by the user, but it certainly can be. It is called as a subroutine within `LTabundR::lta()`.

Usage

```
g0_combine(g0s, g0cvs)
```

Arguments

<code>g0s</code>	A numeric vector of weighted estimates of trackline detection probability, $g(0)$. Any length is allowed.
<code>g0cvs</code>	A numeric vector, the same length as <code>g0s</code> , of weighted estimates of the CV of $g(0)$.

Value

A list with two slots: `$g0` holds the combined $g(0)$ estimate, and `$CV` holds the combined CV.

g0_model	<i>Estimate relative g(0) in different survey conditions</i>
----------	--

Description

This function is an implementation of Barlow (2015), "Inferring trackline detection probabilities, $g(0)$, for cetaceans from apparent densities in different survey conditions" (*Marine Mammal Science*), for processed Wincruz survey data. This function predicts the relative $g(0)$ (compared to Beaufort sea state 0) for all Beaufort sea states 0 - 6.

Usage

```
g0_model(
  spp,
  truncation_distance = 5.5,
  cruz,
  cohort = 1,
  eff_types = "S",
  jackknife_fraction = 0.1,
  seed = NULL,
  constrain_shape = FALSE,
  k = 4,
```

```

    toplot = TRUE,
    verbose = TRUE
  )

```

Arguments

spp	A character vector of species code(s) whose relative trackline detection probability ($g(0)$) you want to estimate.
truncation_distance	The truncation distance, in km, to apply to sightings.
cruz	Your <code>cruz</code> object (produced from <code>LTabundR::process_surveys()</code>). Ensure that segment lengths are short; Barlow (2015) used 10km segments. See using the built-in <code>LTabundR</code> dataset, <code>data(noaa_10km_1986_2020)</code> , if it applies to your study.
cohort	The cohort whose data pertains to the species of interest, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.
eff_types	Effort types to filter segments and sightings to before conducting analysis. The default is systematic effort only ("S"). Can be NULL.
jackknife_fraction	The proportion of data to leave out within each jackknife permutation, which is used to calculate the CV of the $Rg(0)$ estimates. The default is 0.1 (i.e., 10% of the data, yielding 10 jackknife loops), after Barlow (2015).
seed	Set a seed (any integer) to ensure that the result is reproducible. If left NULL, the results are liable to differ slightly for each run of this function.
constrain_shape	Some $Rg(0)$ curves will not decline monotonically due to sample size issues at low Bft (0-2) or high Bft (5-6) states. To coerce monotonic decline, set this to TRUE, and the function will use a shape-constrained GAM (<code>scam()</code> from package <code>scam</code>) instead of a classic <code>mgcv::gam()</code> .
k	Smoothing term for the Bft spline in the GAM. Default (and the value used in Barlow 2015) is 4.
toplot	Boolean, with default TRUE, indicating whether segment length histograms and detection function plots (<code>Distance::plot.ds()</code>) should be displayed (during estimation of effective strip width).
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Details

After Barlow (2015), this function implements the following procedure:

1. Filter sightings to the specified species code(s), and associate those sightings with their respective 10-km segment ID.
2. Determine the presence or absence of the species of interest for each segment.

3. Estimate the Effective Search Area (ESA) for each Beaufort sea state represented in the data. This is done by fitting a detection function to the sightings data (half-normal key; using function `Distance::ds()`), using Beaufort as the only covariate. With this detection function model, the probability of detection ($p(\text{det})$) for sightings in each Beaufort state is determined (`Distance::predict.ds()$fitted`). The effective strip width (ESW) for each Beaufort state is determined by multiplying $p(\text{det})$ by the supplied `truncation_distance`. The ESW is used to calculate the ESA in each segment ($\text{ESA} = 2 \times L \times \text{ESW}$). Note that Beaufort states 0 and 1 are combined (after Barlow 2015), since they are typically under-represented in open-ocean fieldwork.
4. A binomial Generalized Additive Model (GAM) ("logit" link function) is fit to the segments (using `mgcv::gam`), based upon Equation 3 from Barlow (2015), which includes Beaufort sea state, Latitude x Longitude, and Year as predictors (using the `mgcv::s()` spline function) and the log of ESA as an offset (`stats::offset()`) to predict the probability of apparent presence in each segment. To prevent over-fitting, the spline functions for Beaufort and Year are constrained with $k=4$, and model complexity penalty is inflated ($\text{gamma} = 1.4$ in `mgcv::gam()`).
5. The GAM model is used to predict the probability of apparent species presence (`mgcv::predict.gam()`) in fake segments of equal length, year, and Latitude/Longitude but with different Beaufort sea states, 0 - 6.
6. By comparing the probability for Beaufort 0 (in which rates of apparent presence would be highest) to that for other Beaufort states, the Relative $g(0)$ ($Rg(0)$) is estimated (Equation 4 in Barlow 2015).
7. The Coefficient of Variation (CV) for $Rg(0)$ is then estimated using a jackknife procedure, in which a fraction of the data are sequentially removed (input `jackknife_fraction`, with default 10%), $Rg(0)$ is re-estimated by implementing steps 4 - 6 above, and this process is repeated for all sequential fractions of the data. The CV is derived from these pseudo-estimates (let's call them jacks) with the equation $n \times \text{CV}(\text{jacks}) - (n-1) \times \text{CV}(\text{jacks})$, in which n is the number of jackknife estimates ($1 / \text{jackknife_fraction}$) and CV is the standard deviation of jacks divided by their mean.

Value

A list:

- `Rg0` a data.frame of estimates of Relative $g(0)$ at each Beaufort.
- `gam` the `mgcv::gam()` model output.
- `jackknife` A list of jackknife results: `$g0` contains the $g(0)$ estimates from each jackknife iteration; `$ESW` contains the effective half-strip-width estimates; `$gams` contains the gam model objects.
- `summary`: A dataframe summarizing results; **this is the primary output you are likely to use.**
- `cruz10`: A modified nascent `cruz` object that has been re-segmentized to have segments of 10km-length. This usually takes a while to create, so this output gives you the option of passing this object on to your next call of `g0_bft_model()` (see `cruz10` argument above) to save time.

g0_optimize

*Find g0 values Nelder-Mead using optimization***Description**

This is an internal function used in `LTabundR::lta()` for finding the values (through `optim()`) for the parameters needed to conduct parametric bootstrapping of g_0 values for each non-parametric bootstrap of the LTA routine.

Usage

```
g0_optimize(g0, g0_cv, try_count = 20, seed = NULL, verbose = TRUE)
```

Arguments

<code>g0</code>	Initial estimate of $g(0)$.
<code>g0_cv</code>	Initial estimate of $g(0)$ coefficient of variation.
<code>try_count</code>	Number of times to attempt optimization before giving up.
<code>seed</code>	Set a seed (any integer) to ensure that the result is reproducible. If left <code>NULL</code> , the results are liable to differ for each run of this function.
<code>verbose</code>	Print updates to the Console?

Value

A list with three named slots: `[[1]] g0_mean`, the exact mean, according to the optimization results; `[[2]] g0_cv`, the CV according to optimization; and `[[3]] bestFit`, a numeric vector of length 2 containing parameters to pass to the parameteric $g(0)$ bootstrap routine in `LTabundR::lta()`. If the initial $g(0)$ estimate is 1.0, or the initial $g(0)$ CV estimate is 0.0, then these parameters will be returned as `NA`.

g0_plot

*Plot $Rg(0)$ curves***Description**

This function produces a multi-pane plot of changes in relative trackline detection probabilities, $Rg(0)$, across Beaufort sea states for a set of species.

Usage

```
g0_plot(Rg0, panes = 3, legend_key = 0.4, legend_font = 7)
```

Arguments

Rg0	A data.frame of Rg(0) estimates for a set of species groups, as produced by the LTabundR function g0_table(). See its documentation for details.
panes	The number of plot panes to produce. Species will be assigned to each pane according to their mean Rg(0) estimate, such that species with similar detectability will be grouped together.
legend_key	Height of each legend key, in cm, to allow for fine-tuning the formatting of the plot.
legend_font	Legend font size, to allow for fine-tuning the formatting of the plot.

Value

A ggplot object with faceted panes arranged vertically.

<code>g0_results</code>	<i>Relative trackline detection probabilities, Rg(0)</i>
-------------------------	--

Description

Relative trackline detection probabilities, $Rg(0)$

Usage

`g0_results`

Format

An object of class `data.frame` with 175 rows and 13 columns.

Details

$Rg(0)$ estimates for 25 species groups, based on survey data from 1986-2020 (see `noaa_10km_1986_2020` dataset).

g0_table	<i>Compile table of Relative $g(0)$ estimates by Beaufort state</i>
----------	--

Description

This function estimates relative trackline probabilities, $Rg(0)$, for a set of species and returns a single table (a `data.frame`) with the results for each species. This function is a wrapper for `LTabundR::g0_model()`. See its documentation for details on the procedure.

Usage

```
g0_table(
  cruz,
  species,
  eff_types = "S",
  jackknife_fraction = NULL,
  seed = NULL,
  topplot = TRUE,
  verbose = TRUE
)
```

Arguments

cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>). Ensure that segment lengths are short; Barlow (2015) used 10km segments.
species	A list of sublists, in which each sublist details the settings for a single species (or related group of species). Each sublist needs to have these slots: • <code>spp</code>: A character vector of species code(s) whose relative trackline detection probability ($g(0)$) you want to estimate. • <code>title</code>: A unique title, such as a common name, scientific name, or genus, to easily summarize this species group. • <code>cohort</code>: The cohort whose data pertains to the species of interest, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced. If not provided, assumed to be 1. • <code>truncation</code>: The truncation distance, in km, to apply to sightings. • <code>constrain_shape</code> Some $Rg(0)$ curves will not decline monotonically due to sample size issues at low Bft (0-2) or high Bft (5-6) states. To coerce monotonic decline, set this to <code>TRUE</code>, and the function will use a shape-constrained GAM (<code>scam()</code> from package <code>scam</code>) instead of a classic <code>mgcv::gam()</code>. • <code>k</code> Smoothing term for the Bft spline in the GAM. Default (and the value used in Barlow 2015) is 4. • <code>regions</code>: A way to specify that cruz data should be filtered to a certain geostratum region before conducting the analysis. If not <code>NULL</code>, this input must match one of the geostratum names in <code>cruz\$strata</code>.
eff_types	Effort types to filter segments and sightings to before conducting analysis. The default is systematic effort only ("S"). Can be <code>NULL</code> .

jackknife_fraction	The proportion of data to leave out within each jackknife permutation, which is used for estimating the CV of $Rg(0)$ estimates. The default is 0.1 (i.e., 10% of the data, yielding 10 jackknife loops), after Barlow (2015).
seed	Set a seed (any integer) to ensure that the result is reproducible. If left NULL, the results are liable to differ for each run of this function.
topplot	Boolean, with default TRUE, indicating whether segment length histograms and detection function plots (<code>Distance::plot.ds()</code>) should be displayed (during estimation of effective strip width).
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Value

A data.frame, in which every row details the $Rg0$ estimate for a single species group in a single Beaufort sea state, with the following columns:

1. title: The title given to the species group.
2. spp: Species codes, concatenated by a hyphen if there are multiple in the species group.
3. bft: Beaufort sea state.
4. $Rg0$: Estimate of the relative $g(0)$ for this sea state.
5. ESW: Estimate of the effective strip half-width (ESW) for this sea state.
6. $Rg0_SE$: The standard error of the $Rg(0)$ estimate, non-zero only if jackknife_fraction was used.
7. $Rg0_CV$: The CV of the $Rg(0)$ estimate, non-zero only if jackknife_fraction was used.
8. ESW_SE : The standard error of the ESW estimate, non-zero only if jackknife_fraction was used.
9. sits: The number of sightings within this sea state used to model $Rg(0)$.
10. sits_p: The proportion of sightings within this sea state.
11. segs: The number of segments within this sea state used to model $Rg(0)$.
12. segs_p: The proportion of segments within this sea state.

See the built-in dataset, `data(g0_results)`, for an example of the output.

<code>g0_weighted</code>	<i>Find a single weighted $g(0)$ estimate (and CV) from a set of Beaufort-specific estimates.</i>
--------------------------	--

Description

Use this function to summarize the Beaufort-specific model of relative trackline probabilities (from `LTabundR::g0_bft_model()`) as a single relative $g(0)$ estimate, weighted by the prevalence of each Beaufort sea state within the survey data. This function automatically estimates the CV of this weighted value using an iterative loop with an MCMC core, adapted from Jeff Moore's code.

Usage

```

g0_weighted(
  Rg0,
  Rg0_cv,
  cruz,
  cohort = 1,
  iterations = 10000,
  seed = NULL,
  beta_range = c(-1.5, 0),
  beta_step = 0.001,
  beta_sd_range = c(0, 0.3),
  beta_sd_step = 1e-04,
  ymax = 0.2,
  calculate_pars = TRUE,
  topplot = TRUE,
  verbose = TRUE
)

```

Arguments

Rg0	A numeric vector, length 7, of Relative $g(0)$ estimates for Beaufort sea states 0 - 6. See documentation of <code>LTabundR::g0_bft_model()</code> , which provides this vector as a column in its output, for details on the concept of relative trackline detection probabilities.
Rg0_cv	A numeric vector, length 7, of the CV of Relative $g(0)$ estimates. See same documentation as above for details.
cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).
cohort	The cohort name within your cruz object that you want to analyze.
iterations	The number of draws used to simulate the underlying distribution.
seed	Set a seed (any integer) to ensure that the result is reproducible. If left NULL, the results are liable to differ for each run of this function.
beta_range	The range in distribution parameters for simulating the $g(0)$ mean.
beta_step	The interval between candidate values within the above range. A smaller step means higher resolution and accuracy, but takes longer to process.
beta_sd_range	The range in distribution parameters for simulating the $g(0)$ SD.
beta_sd_step	The interval between candidate values for the above SD range.
ymax	The maximum value on the y axis of the diagnostic Rg0 SD plot.
calculate_pars	Boolean with default TRUE; an option to skip estimation of plogis distribution parameters, to save time.
topplot	Boolean, with default TRUE, indicating whether results should be plotted.
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Details

This code is an adaptation and expansion of Jeff Moore's code (February 2020). This function automatically estimates the CV of this weighted value using an iterative loop with an MCMC core. The MCMC routine is described in Bradford et al. (2021). The iterative loop tries various candidate values for the distribution parameters and calculates the cumulative error of the resulting distribution at each Beaufort state. It then determines the candidate value with the lowest error, weighting each error according to the proportion of effort occurring in the respective Beaufort state.

Value

The output is provided in the form of a list with two slots: `g0` and `bft`. The `g0` slot holds a data.frame in which `$wt.mean` is the weighted mean $g(0)$ value across all Beaufort states for the survey data provided, and `$wt.cv` is the CV of that estimate. Other values are reported to facilitate QA/QC, but `$wt.mean` and `$wt.cv` is what you would pass to an abundance estimation routine, such as `LTaundR::lta()`. The `bft` slot holds a dataframe with the proportion of effort in each Beaufort sea state.

<code>gg_effort</code>	<i>Add cruz effort to a base ggplot2 map.</i>
------------------------	---

Description

Add cruz effort to a base ggplot2 map.

Usage

```
gg_effort(
  p,
  cruz,
  cohort = 1,
  color_by_bft = FALSE,
  color = "black",
  lwd = 0.1,
  alpha = 0.5
)
```

Arguments

<code>p</code>	Base ggplot2 to which you are adding this geom.
<code>cruz</code>	Your cruz object (produced from <code>LTaundR::process_surveys()</code>).
<code>cohort</code>	The cohort whose data you would like to map, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.
<code>color_by_bft</code>	Color code by Beaufort sea state?
<code>color</code>	Color string.
<code>lwd</code>	Effort thickness.
<code>alpha</code>	Effort transparency.

Value

An updated ggplot2 object.

gg_geostratum	<i>Add cruz geostrata to a base ggplot2 map.</i>
---------------	--

Description

Add cruz geostrata to a base ggplot2 map.

Usage

```
gg_geostratum(  
  p,  
  cruz,  
  strata,  
  color = "orchid4",  
  lwd = 0.5,  
  lty = 1,  
  crs = 4326  
)
```

Arguments

p	Base ggplot2 to which you are adding this geom.
cruz	Your cruz object (produced from LTabundR: :process_surveys()).
strata	desc
color	Color string.
lwd	Boundary line width.
lty	Boundary line type.
crs	Optional CRS.

Value

An updated ggplot2 object.

`gg_sightings`*Add cruz sightings to a base ggplot2 map.*

Description

Add cruz sightings to a base ggplot2 map.

Usage

```
gg_sightings(  
  p,  
  cruz,  
  cohort = 1,  
  spp = NULL,  
  spp_translate = NULL,  
  color_by_spp = FALSE,  
  color = "black",  
  pch = 16,  
  cex = 0.5,  
  alpha = 0.5  
)
```

Arguments

<code>p</code>	Base ggplot2 to which you are adding this geom.
<code>cruz</code>	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).
<code>cohort</code>	The cohort whose data you would like to map, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.
<code>spp</code>	Character vector of species codes.
<code>spp_translate</code>	Dataframe of species code information for translating between codes and common names. See <code>data(species_codes)</code> for an example.
<code>color_by_spp</code>	Color code by species code?
<code>color</code>	Color string.
<code>pch</code>	Sighting point type.
<code>cex</code>	Sighting point size.
<code>alpha</code>	Sighting transparency.

Value

An updated ggplot2 object.

ggplot_idl	<i>Modify a ggplot object to handle the International Date Line (IDL)</i>
------------	---

Description

Modify a ggplot object to handle the International Date Line (IDL)

Usage

```
ggplot_idl(  
  p_base,  
  lon_range = c(100, -100),  
  lat_range = NULL,  
  idl_padding = 0.02,  
  axes = TRUE,  
  bypass = FALSE  
)
```

Arguments

p_base	A ggplot object that you want to visualize in a way that handles the International Date Line.
lon_range	Desired longitude range, as a two-element vector. The first element should be a positive number between 0 and 180 (indicating decimal degrees East), the second should be a negative number between -180 and 0 (indicating decimal degrees West).
lat_range	Desired latitude range, as a two-element vector of decimal degrees (positive = N, negative = S).
idl_padding	The space between the Eastern hemisphere plot and the Western hemisphere plot.
axes	If FALSE, axis labels and ticks will be removed.
bypass	An option to return an unmodified ggplot, to ease automated toggling when using in various functions within LTabundR.

Value

A ggplot2 object that is actually two plots side by side, with a small space in between indicating the IDL.

 grp_calibrate_abund *Group size calibration*

Description

This is an internal function typically not called by a user directly. It is called by the subroutine `group_size()`, which is itself a subroutine of `process_sightings()`.

This function performs school size calibration for a single observer's estimates.

Usage

```
grp_calibrate_abund(
  obs,
  bft,
  yr,
  gbest,
  glow,
  ghigh,
  gs_coefficients = NULL,
  calibrate_floor = 0
)
```

Arguments

<code>obs</code>	Character vector with single observer ID. If this observer ID is contained within <code>gs_coefficients</code> , its specific calibration coefficients will be used. Otherwise the generic calibration coefficients will be used.
<code>bft</code>	Numeric of Beaufort sea state during sighting.
<code>yr</code>	Numeric of year of sighting.
<code>gbest</code>	Numeric of best school size estimate.
<code>glow</code>	Numeric of low school size estimate.
<code>ghigh</code>	Numeric of high school size estimate.
<code>gs_coefficients</code>	Dataframe of school size coefficients. If not provided, calibration will not be attempted.
<code>calibrate_floor</code>	This argument accepts a number indicating the minimum raw school size estimate for which school size calibration will be attempted.

Value

A one-row dataframe with the best school size estimate, the variance of the estimate, a boolean indicating whether calibration was possible, and the slope and intercept used for calibration. Passed back to `group_size()`.

grp_coeff_abund	<i>Group size coefficients for the ABUND calibration approach</i>
-----------------	---

Description

Group size coefficients for the ABUND calibration approach

Usage

grp_coeff_abund

Format

An object of class data.frame with 162 rows and 21 columns.

Details

A table of group size calibration coefficient values for a variety of species for a variety of observer codes. This table was provided by Jay Barlow (NOAA NMFS biologist), was created prior to 2016, and has been used for published analyses through 2021 with ABUND versions 7 & 8.

grp_coeff_gerrodetta	<i>Group size coefficients for the Gerrodette calibration approach</i>
----------------------	--

Description

Group size coefficients for the Gerrodette calibration approach

Usage

grp_coeff_gerrodetta

Format

An object of class data.frame with 19 rows and 4 columns.

Details

A table of group size calibration coefficient values for a variety of species for use in the "Gerrodette" calibration approach. These values were drawn from Gerrodette et al. 2019 and used in Barlow et al. (2026).

`grp_ops_abund`*Calibration options for ABUND approach*

Description

Calibration options for ABUND approach

Usage

`grp_ops_abund`

Format

An object of class `list` of length 6.

Details

Example of calibration options to use for the ABUND group-size-calibration approach.

`grp_ops_gerrodette`*Calibration options for Gerrodette approach*

Description

Calibration options for Gerrodette approach

Usage

`grp_ops_gerrodette`

Format

An object of class `list` of length 6.

Details

Example of calibration options to use for the "Gerrodette" group-size-calibration approach.

grp_recalibrate	<i>Recalibrate processed sightings</i>
-----------------	--

Description

Re-run group size calibration for a selection of sightings within a pre-existing cruz object.

Usage

```
grp_recalibrate(
  cruz,
  cohort = 1,
  spp = NULL,
  include_mixed = TRUE,
  mixed_only = FALSE,
  spp_max_only = FALSE,
  ops_max = grp_ops_gerrodette,
  ops_not = grp_ops_abund,
  verbose = TRUE,
  toplot = TRUE
)
```

Arguments

cruz	Your cruz object (produced from LTabundR::process_surveys()).
cohort	The cohort whose sightings you would like to recalibrate, provided as a number indicating which slot in cruz\$cohorts should be referenced.
spp	Character vector of species codes to filter by. Leave NULL if you want to recalibrate all sightings in the cohort.
include_mixed	If TRUE, mixed-species groups containing the spp of interest will be recalibrated, including estimates for non-target species present in those mixed groups.
mixed_only	If TRUE, <i>only</i> mixed-species groups will be recalibrated. Ignored if include_mixed is FALSE.
spp_max_only	If TRUE, the only mixed-species groups recalibrated will be those in which the spp of interest are the most abundant species in the group (i.e., spp_max is found in spp). Ignored if include_mixed is FALSE.
ops_max	The calibration options to apply to the predominant species in the group. Default is the "Gerrodette" method, with default values provided in the built-in dataset data(grp_ops_gerrodette).
ops_not	The calibration options to apply to the non-predominant species in any mixed groups contained in the set of sightings to be recalibrated. Default is the "ABUND" method, with default values provided in the built-in dataset data(grp_ops_abund).
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.
toplot	Boolean, with default TRUE, indicating whether a diagnostics plot should be displayed at the end of the recalibration.

Value

- A list with the following slots:
- 1. key: A data.frame key of which sightings were re-calibrated, with three columns: i = row index from cruz\$cohorts[[cohort]]\$sightings; SightNoDaily = unique Sighting identifier from same; species = the species code.
 - 2. recalibrated_sits: Complete sightings rows but only for the sightings that were recalibrated.
 - 3. updated_cruz: Updated cruz object in which cruz\$cohorts[[cohort]]\$sightings includes the recalibrated sightings as well as all untouched sightings.

grp_size	<i>Group size estimation</i>
----------	------------------------------

Description

This is an internal function typically not called by a user directly. This function is used as a subroutine in process_sightings(). It produces school size estimates for each species within a sighting, based upon all estimates of school size and species composition provided by observers.

Usage

```
grp_size(  
  grp,  
  calibrate = grp_ops_abund,  
  geometric_mean = FALSE,  
  use_low_if_na = TRUE,  
  debug_mode = FALSE,  
  verbose = TRUE  
)
```

Arguments

grp	A dataframe, in which each row is a school size estimate from a single observer, and each column is a column from the DAS dataframe that is relevant to school size estimation (columns Event, year, Bft, and Prob:GsSchoolLow).
calibrate	If NULL, group sizes will not be calibrated. Otherwise supply a list to specify which calibration approach to use. Currently two approaches are supported: that used by ABUND 7/8 (the default shown), which takes an observer-specific, species-agnostic approach, and that developed in Gerrodette et al. (2019) and applied in Barlow et al. (2026), which includes a species-specific, observer-agnostic approach. Details in ?load_cohort_settings().
geometric_mean	This argument accepts a Boolean; if TRUE, geometric means will be calculated instead of arithmetic means. If school size calibration is carried out, the geometric mean will be weighted by calibration variance, such that estimates from observers with low variance will receive more weight. When this function is used

withing `process_sightings()`, this setting from the `cruz` object will be provided. Note that, although only the best estimates may be calibrated if specified above (never the highs and lows), the *same* kind of averaging function *is* applied to the highs and lows as is applied to the bests. That is, when `geometric_mean` is `TRUE`, the geometric mean of the highs and the lows is returned. If the best estimates are calibrated, the geometric *weighted* mean will be applied to the highs and lows, using the variance of the calibrated best estimates as weights. If the best estimates are *not* calibrated, the *unweighted* geometric mean is used to estimate the highs, lows, and bests.

<code>use_low_if_na</code>	If this setting is <code>TRUE</code> , when no observer makes a best estimate of group size, mean group size will be calculated from "low" estimates. This will be done only if no observer has a "best" estimate. When this function is used withing <code>process_sightings()</code> , this setting from the <code>cruz</code> object will be provided.
<code>debug_mode</code>	Boolean, with default <code>FALSE</code> , indicating whether details should be printed to the Console that facilitate debugging.
<code>verbose</code>	Boolean, with default <code>TRUE</code> , indicating whether or not updates should be printed to the Console.

Value

A dataframe in which each row is a species within the sighting, with final estimates of best / high / low and metadata regarding calibration. When this function is used internally by `LTabundR`, the results are passed back to `process_sightings()`.

<code>idl_coerce</code>	<i>Coerce eastern longitudes to western (i.e., less than - 180)</i>
-------------------------	---

Description

This can be useful for plotting datasets that cross the international date line (IDL). (Designed as a behind-the-scences function that most users wont' use, but making it available just in case.)

Usage

`idl_coerce(df)`

Arguments

`df` Dataframe of coordinates with columns `Lon` and `Lat`.

Value

Corrected dataframe (all coordinates between 0 and less than -180).

land	<i>Land</i>
------	-------------

Description

Land

Usage

land

Format

An object of class sf (inherits from data.frame) with 10 rows and 4 columns.

Details

World dataset of land polygons, downloaded from Natural Earth.

load_cohort_settings	<i>Load cohort-specific settings</i>
----------------------	--------------------------------------

Description

This function builds a list of cohort-specific settings – applying only to the analyses for a certain group of species – which you will pass to load_settings().

Usage

```
load_cohort_settings(
  id = "default",
  species = NULL,
  strata = NULL,
  probable_species = FALSE,
  sighting_method = 0,
  cue_range = 0:7,
  group_size_range = c(0, 10000),
  group_size_calibrate = grp_ops_abund,
  use_low_if_na = FALSE,
  io_sightings = 0,
  geometric_mean_group = TRUE,
  truncation_km = 20,
  beaufort_range = 0:6,
  abeam_sightings = FALSE,
  strata_overlap_handling = "smallest",
  distance_types = c("S", "F", "N"),
```

```

    distance_modes = c("P", "C"),
    distance_on_off = TRUE
)

```

Arguments

id	An informal identifier for this cohort, to help you keep track of which cohort is which. For example, settings for a cohort of large whales species could be named "big_whales"; settings for small delphinids and phocoenids could be named "small_odontocetes"; settings for beaked whales could be named "beakers".
species	A character vector of species codes to include in this cohort. If NULL (the default), all species will be included.
strata	A character vector of geostratum names; these must match the names listed in the strata slot of your survey settings (see documentation for load_settings()). If NULL (the default), <i>all</i> geostrata in your survey settings will each be used. This argument is an opportunity to subset the geostrata used for a cohort; for example, certain dolphin species in Hawaiian waters have unique geostrata that should apply to them but not to other species.
probable_species	If TRUE (default is FALSE), the "probable" species identifications will be used in place of the "unidentified" categories.
sighting_method	A coded integer which determines which sightings will be included based on how they were first seen. Allowable codes are 0=any method, 1=with 25X only, 2=neither with 25x binoculars nor from the helicopter (i.e., naked eyes and 7x binoculars only). These codes match those used in ABUND7/9.
cue_range	Numeric vector of acceptable "observation cues" for sightings used in estimates of abundance. (0=this detail is missing in the data, 1=associated birds, 2=splashes, 3=body of the marine mammal, 4=associated vessel, 5=?, 6=blow / spout, 7=associated helicopter). These codes match those used in ABUND7/8/9.
group_size_range	Minimum and maximum group sizes to be included in estimates of abundance. This is the overall group size, not the number of the given species that are present in a mixed group.
group_size_calibrate	If NULL, group sizes will not be calibrated. Otherwise supply a list to specify which calibration approach to use. Currently two approaches are supported: that used by ABUND 7/8 (the default shown), which takes an observer-specific, species-agnostic approach, and that developed in Gerrodette et al. (2019) and applied in Barlow et al. (2026), which includes a species-specific, observer-agnostic approach. See Details below.
use_low_if_na	If this setting is TRUE, and if an observer does not make a best estimate of group size, mean group size will be calculated from "low" estimates. This will be done only if no observer has a "best" estimate.
io_sightings	A coded integer which specifies how sightings by the independent observer will be handled. Allowable codes, which are inherited from those used in ABUND7/9,

are "-1"=include independent observer sightings with all other sightings, "0"=ignore sightings by independent observer, "1"=use only sightings made by regular observer team WHEN an independent observer was present, "2"=include only sightings made by the independent observer. IO sightings are typically used only for making $g(0)$ estimates, otherwise IO sightings are usually ignored (code = "0").

geometric_mean_group

This argument accepts a Boolean; if TRUE, geometric mean school sizes will be calculated instead of arithmetic means. Also, if school size calibration is carried out, the geometric mean will be weighted by calibration variance, such that estimates from observers with low variance will receive more weight. Barlow, Gerrodette, and Perryman (1998) found that using the geometric mean yielded slightly better performance than a simple arithmetic mean group size.

truncation_km Specifies the maximum perpendicular distance for groups that are to be included for abundance estimation. Also determines the bins used for grouped perpendicular distances.

beaufort_range A numeric vector indicating the Beaufort sea states (0 - 7) to be accepted within usable segments.

abeam_sightings

If TRUE, sightings that occur aft of beam are included in estimating the detection function and densities. Default is FALSE: all abeam sightings will not be used in density estimation or detection function estimation.

strata_overlap_handling

This setting informs how effort is split into segments when surveys cross stratum boundaries, and also which stratum name is assigned to each row of data. See Details below.

distance_types A character vector of the full-range of effort types that meet the "analysis inclusion criteria", i.e., will be included in detection function estimation, and therefore considered in effort segmentizing. Accepted values are "S" (systematic/standard effort), "F" (fine-scale effort), and "N" (non-systematic/non-standard effort, in which systematic protocols are being used but effort is not occurring along design-based transect routes). Note that it is possible to further subset the effort types specifically for density estimation in LTabundR's function for line transect analysis, `lta()`.

distance_modes The effort modes that meet "analysis inclusion criteria", i.e., will be included in detection function estimation, and therefore considered in effort segmentizing. Accepted values are "P" (passing) and "C" (closing)

distance_on_off

The value(s) of OnEffort (On Effort is TRUE, Off Effort is FALSE) that will be included in detection function estimation, and therefore considered in effort segmentizing.

Details

More information on the list required for the `group_size_calibrate` input: The list requires the following fields (examples are given in the built-in datasets `grp_ops_abund` and `grp_ops_gerrodetete`):

- `method`: accepted values = 'ABUND' and 'Gerrodette';
- `coefficients`: a `data.frame`. For the "ABUND" method, this is a table of observer-specific calibration coefficients. To use the same coefficients that have been in use at SWFSC and PIFSC up to 2021, see the built-in dataset `data(grp_coeff_abund)`. For the "Gerrodette" method, this is a table of species-specific coefficients. To use the same coefficients from the method's originating papers, see the built-in dataset `data(grp_coeff_gerrodette)`. For both methods, supplied `data.frame`'s must match the column naming structure of the respective built-in dataset.
- `floor`: minimum group size to which calibration will be applied. In the "ABUND" approach, this is referenced only for observers who do not have an entry in the `coefficients` table, which has a calibration floor for each observer. The default is 0, meaning that calibration will be attempted for *all* school size estimates, regarding of the raw estimate.
- `intercept` (ignored if not using the "Gerrodette" method) the value to use for the "intercept" coefficient if a species in this cohort is not present in the `coefficients` table.
- `beta` (ignored if not using the "Gerrodette" method): the value to use for the "beta" coefficient if a species in this cohort is not present in the `coefficients` table.
- `beta_mixed` (ignored if not using the "Gerrodette" method): used for mixed-species groups. If NULL, the "beta" coefficient for the most abundant species in the group will be used. If a numeric value is supplied here, that value will be used instead.

Note that the high and low estimates are *never* calibrated; only the best estimates are.

More information on the list required for the `strata_overlap_handling` input: Note that the main impact of this setting is on how effort is broken into segments; the assigned stratum name is for display only and will not constrain options for including/excluding strata in analyses farther along in the LTabundR workflow.

- The default option is "smallest", which means that effort will always be assigned to the smallest stratum when multiple strata overlap spatially. This is a safe option for surveys with "nested" strata (such as the Central North Pacific strata used by NOAA Fisheries).
- Another option is "each" in which each time a stratum boundary is crossed the current segment will end and a new segment will begin. Also, stratum assignments for each row of effort will be shown as a concatenation of all the stratum layers overlapping at its position (e.g., "OtherCNP&HI_EEZ"). Note that the "each" option segmentizes effort in the exact same way as "smallest" when strata are fully nested; its main advantage is in dealing with partially overlapping strata.
- The third option is "largest", in which the largest of overlapping strata is used to assign a stratum name to each row. (We are not sure what use case this would serve, but we offer it as an option for niche analyses.)

Value

A list with named slots, equivalent to your input arguments. Save this output to an object, e.g., using the same name you provided in the `id` argument, and pass it to `load_settings()`.

load_settings

Load settings for processing WinCruz surveys

Description

You need the output of this function to pass to `process_surveys()`, the primary function for processing WinCruz data.

Usage

```
load_settings(strata = NULL, survey = NULL, cohorts = NULL)
```

Arguments

- | | |
|---------|---|
| strata | A named list in which each slot is a <code>data.frame</code> of coordinates for a geostratum polygon. Each <code>data.frame</code> must have <code>Lon</code> and <code>Lat</code> as the first two columns, providing coordinates in decimal degrees in which West and South coordinates are negative. It is acceptable if vertices in the eastern hemisphere are described using negative longitudes below -180, e.g., -185. Other columns are allowed, but the first two need to be <code>Lon</code> and <code>Lat</code> . The name of the slot holding the <code>data.frame</code> will be used as a reference name for the stratum. For an example of formatting, see <code>data(strata_cnp)</code> . If you are working with a standard NOAA survey region, such as the Central North Pacific (CNP), Eastern Tropical Pacific (ETP), or California Current System (CCS), you can use built-in polygons available in <code>data(strata_cnp)</code> , <code>data(strata_etp)</code> , or <code>data(strata_ccs)</code> , respectively. To explore and/or select strata contained within those built-in datasets, use the functions <code>strata_explore()</code> and <code>strata_select()</code> . Note that if coordinates in your data or in your collection of strata span the International Date Line (IDL) such that some longitudes are positive and some are negative, during data processing all longitudes will be coerced to negative degrees West. Also note that area calculations for strata that span the international date line may not be accurate in the current version of <code>LTabundR</code> . |
| survey | Survey-wide settings, provided as the list that is generated with the command <code>load_survey_settings()</code> . See the documentation for that function for full details. If nothing is supplied, the default survey-wide settings (see <code>load_survey_settings()</code>) will be applied. |
| cohorts | Cohort-specific settings, provided as a list of lists. Each slot contains the settings for a single cohort, which can be generated with the command <code>load_cohort_settings()</code> . See the documentation for that function for full details. Cohort-specific settings apply only to a group of species. Since you can add as many cohorts to a settings object as you need, this allows you to stage your entire analysis and run your code once, without modifying code between the analysis of each cohort. If nothing is supplied, the default cohort settings (see <code>load_cohort_settings()</code>) will be applied to all species. |

Value

A list with three named slots, equivalent to your three input arguments. Save this output to an object, e.g., "settings", and pass it to `process_surveys()`.

load_survey_settings	<i>Load survey settings</i>
----------------------	-----------------------------

Description

This function builds a list of survey-wide settings (applying to all segments and all sightings of all species, regardless of their cohort designation), which you will pass to `load_settings()`.

Usage

```
load_survey_settings(
  out_handling = "remove",
  interpolate = NULL,
  min_row_interval = 2,
  max_row_interval = 3600,
  max_row_km = 100,
  km_filler = 1,
  speed_filler = 10 * 1.852,
  segment_method = "day",
  segment_target_km = 150,
  segment_max_interval = 48,
  segment_remainder_handling = "segment",
  seed = NULL,
  ship_list = NULL,
  species_codes = NULL,
  smear_angles = FALSE
)
```

Arguments

<code>out_handling</code>	This argument allows you to specify how data occurring outside of geo-strata should be handled. If this is set to "remove", those rows will be filtered out of the data early in the process. This reduces memory usage, speeds up processing, and gives you geographic control of how effort and sightings will be summarized. If this is set to "stratum", those data will be assigned to a fake geo-stratum, named "out". Effort in the "out" stratum will not be segmentized, but "out" sightings will be processed and retained in the final datasets. This setting might be useful if you want to use "out" data for survey summaries and/or detection function estimation. The default is "remove", since that saves the most time and memory. If no geostratum is provided, this setting will be ignored and all rows of data will be assigned to a stratum called "none".
---------------------------	--

<code>interpolate</code>	This argument allows you to interpolate the DAS data at the onset of processing if your position updates are separated by large time intervals, which would make spatial effort and stratum assignments less exact. If this argument is NULL, then no interpolation will occur. If it is a number, e.g., 30, LTabundR will interpolate the data using simple-linear methods (i.e., no great-circle calculations) such that position updates occur every 30 seconds or less. If adjacent DAS rows are from different dates or cruises, the interpolation routine will skip to the next pair of related rows. Interpolation will only occur for On-Effort rows.
<code>min_row_interval</code>	The minimum time interval, in seconds, between rows in order for the Great Circle distance between rows to be calculated. Intervals less than this number will be assigned a distance of 0 km.
<code>max_row_interval</code>	The maximum allowable time interval, in seconds, between rows before LTabundR assumes that there has been a break in survey data logging. The default of 3600 seconds (6 hours) was chosen because there is usually at least 6 hours of night-time darkness between the end of effort on one day and the start of effort on the next.
<code>max_row_km</code>	The maximum allowable distance interval, in km, between rows before the function assumes that there has been a break in survey data logging. The default was chosen arbitrarily to find a value that replicates the processing results from ABUND.
<code>km_filler</code>	When valid speed and position information is not available (e.g., the given distance exceeds <code>max_km_gap</code>), this value (in km) will be used as an estimate of the distance in between consecutive rows of data. The default was chosen arbitrarily to find a value that replicates the processing results from ABUND.
<code>speed_filler</code>	When speed is not available in the data, this value (in kph) will be used as a filler in order to estimate the distance between consecutive rows of data based on timestamp differences (when lat/long coordinates are not available). The default was chosen arbitrarily to find a value that replicates the processing results from ABUND.
<code>segment_method</code>	The two method options are "day" – all effort within the same Cruise-StudyArea-Stratum-Year-Effort scenario (i.e., an effort bloc) will be binned into segments by calendar date – and "equallength" – effort within each unique effort bloc will be divided into segments of approximately equal length.
<code>segment_target_km</code>	If segmentizing by "equallength", this field allows you to specify what that target length is, in km. If segmentizing by "day", this argument is ignored. The default is 150 km, the distance generally surveyed in one day on NOAA Fisheries surveys.
<code>segment_max_interval</code>	A number indicating hours. See Details.
<code>segment_remainder_handling</code>	A character string. See Details.
<code>seed</code>	Set a seed (any integer) to ensure that your survey is processed reproducibly: namely, segments will be chopped the exact same way every time. Some of the

	segment remainder handling methods (namely "segment" and "append") will place the remainder to a randomly selected segment. Supplying a number here will ensure the remainder goes in the same place with each run. If left NULL, the segment breaks are liable to differ each time this function is run, and the segments to which sightings are assigned are liable to vary as well.
ship_list	A data.frame containing a list of ship names. If not provided the default version, which was current as of the release of ABUND9 in 2020, will be used (data(ships)). Supplied data.frames must match the column naming structure of data(ships).
species_codes	A data.frame containing species codes. This is an optional input, chiefly used to format species names in the reporting stage of the workflow (lta_report()) especially). If the user supplies a data.frame it must match the column naming structure of data(species_codes).
smear_angles	If TRUE (the default is FALSE), bearing angles to a group of animals will be "smeared" by adding a uniformly distributed random number between -5 and +5 degrees. This has not been used in any recent analyses because observers have not been rounding angles as much as they used to, according to the release notes for ABUND9. It was suggested by Buckland as a method for dealing with rounding, which is especially influential when rounding to zero places many sightings at zero perpendicular distance.

Details

More information on the `segment_max_interval` input: If segmentizing by "equallength", this setting allows you to specify the time gaps in effort that are allowed to be contained within a single segment. For example, if your goal is a few large segments of equal length (e.g., 150-km segments, for bootstrap estimation of density variance), you are probably willing for discrete periods of effort to be concatenated into a single segment, even if the gaps between effort are as large as 1 or 2 days, in which case you would set `segment_max_interval` to 24 or 48 (hours), respectively. However, if your goal is many smaller segments (e.g., 5-km segments, for habitat modeling), you want to ensure that effort is contiguous so that segment locations can be accurately related to environmental variables, in which case you would set `segment_max_interval` to be very small (e.g., 0.2 hours, or 12 minutes). Setting this interval to a small number, such as 0.2, also allows the segmentizing function to overlook momentary breaks in effort, such as when an unofficial observer logs a sighting. If segmentizing by "day", this argument is ignored.

More information on the `segment_remainder_handling` input: If segmentizing by "day", this argument is ignored. But if segmentizing by "equallength", periods of effectively-contiguous effort (as specified by `segment_max_interval`) are unlikely to be perfectly divisible by your `segment_target_km`; there is going to be a remainder. You can handle this remainder in three ways:

1. "disperse" allows the function to adjust `segment_target_km` so that there is in fact no remainder, effectively dispersing the remainder evenly across all segments within that period of contiguous effort;
2. "append" asks the function to append the remainder to a randomly selected segment, such that most segments are the target length with the exception of one longer one;
3. "segment" asks the function to simply place the remainder in its own segment, placed randomly within the period of contiguous effort.

This setting also has a second layer of versatility, because it can accept a one- or two-element character vector. If a two-element vector is provided (e.g., `c("append", "segment")`), the first element will be used in the event that the remainder is less than or equal to half your `segment_target_km`; if the remainder is more than half that target length, the second element will be used. This feature allows for replication of the segmentizing methods in Becker et al. (2010).

Value

A list with named slots, equivalent to your input arguments. Save this output to an object, e.g., "survey_settings", and pass it to `load_settings()`.

lta

Line transect analysis

Description

For a single species or species pool, fit a detection function and estimate density / abundance, with an option to conduct parametric / non-parametric bootstrap sampling for variance estimation.

Usage

```
lta(
  cruz,
  Rg0,
  fit_filters,
  df_settings,
  estimates,
  use_g0 = TRUE,
  abund_eff_types = c("S"),
  abund_bft_range = 0:6,
  bootstraps = 0,
  seed = NULL,
  max_attempts = 5,
  results_file = NULL,
  topplot = TRUE,
  verbose = TRUE
)
```

Arguments

cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>). Ensure that this cruz object is filtered only to the years, regions, and sighting conditions you would like to use for detection function fitting. Filter your cruz object with full flexibility using <code>LTabundR::filter_cruz()</code> . Note that filtering for detection function fitting is typically less stringent than filtering for downstream steps for abundance estimation, since as many sightings are included as possible to combat low sample sizes, as long as sightings were observed using standard
------	---

methods in an unbiased search pattern, and as long as you do not expect detectability to vary across years and regions.

<code>Rg0</code>	The result of <code>LTabundR::g0_model()</code> , which is a <code>data.frame</code> with Relative trackline detection probabilities, $g(0)$, for each species in each Beaufort sea state. See <code>LTabundR</code> dataset <code>data(g0_results)</code> as an example. If not provided, weighted $g(0)$ will not be estimated in this function call, and unless you manually provide $g(0)$ values within your <code>estimates</code> input, $g(0)$ will be assumed to be 1 and $g(0)_{cv}$ will be assumed to be 0.
<code>fit_filters</code>	A named list, with filters and settings pertaining to the data used to fit a detection function model. The slots below are recognized, but only <code>spp</code>, <code>cohort</code>, and <code>truncation_distance</code> are required (i.e., do not have defaults). • <code>spp</code>: A character vector of species codes. Using multiple species codes may be useful when you have low sample sizes for a cohort of similar species. • <code>pool</code>: A character string, providing a title for this species pool. If not specified, the species codes used will be concatenated to produce a title automatically. • <code>cohort</code>: The cohort containing these species, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced. • <code>truncation_distance</code>: The truncation distance to apply during model fitting. • <code>other_species</code>: A character vector with four recognized values: – If "apply" (the default if not specified), the species code will be changed to "Other" for sightings in which the species was in a mixed-species school but was not the species with the largest percentage of the total school size. In those cases, the species was not as relevant to the detection of the school as the other species were, which may bias the detection function. This creates a factor level for the detection function to use (when "species" is a covariate) to distinguish between cue-relevant species that are within the specified pool and those that are not. – The second option for <code>other_species</code> is "ignore", which does not reassign species codes to "Other", and ignores whether the species of interest held the plurality for a mixed species detection. – The third option is "remove": any species re-assigned to "Other" will be removed before the detection function is fit; this can be useful if only a small number of species are re-assigned to "Other", which would then obviate species as a viable covariate (since the sample size of all species levels would be unlikely to exceed <code>df_settings\$covariates_n_per_level</code> – see below). – The fourth and final option is <code>coerce</code>, which forces <i>all</i> species codes to "Other" for the purposes of detection function fitting and abundance estimation. This can be useful if you want to toggle the use of species as a covariate for a specific species pool, and/or produce abundance estimates for unidentified taxa (e.g., an 'Unidentified dolphins' species pool that includes multiple species codes).
<code>df_settings</code>	A named list, with parameters for fitting the detection function. The following slots are recognized, but none is required (i.e., all have defaults):

- **covariates:** Covariates you wish to include as candidates in detection function models, provided as a character vector. The covariates must match columns existing within `cruz$cohorts$<cohort_name>$sightings`. Common covariates that you will find within sightings include `c('Bft', 'LnSsTot', 'Cruise', 'Year')`. Note that the function will ignore case, coercing all covariates to lowercase. If `LnSsTot` is included as a covariate, the function will (1) check to see if the sightings dataframe has a column named `ss_valid` (all `cruz` objects do), then, if so, (2) filter sightings only to rows where `ss_valid` is `TRUE`, meaning the school size estimate for that sighting is a valid estimate.
- **covariates_factor:** A Boolean vector, which must be the same length as `covariates`, indicating whether each covariate should be treated as a factor instead of a numeric.
- **covariates_levels:** The minimum number of levels a factor covariate must have in order to be included as an eligible covariate.
- **covariates_n_per_level:** The minimum number of observations within each level of a factor covariate. If this condition is not met, the covariate is excluded from the candidates.
- **simplify_cue** A Boolean, with default `TRUE`, indicating whether or not cue codes should be simplified before being included as a covariate in detection function fitting. This can help to overcome the factor sample size limitations that may prevent inclusion as a covariate. If `TRUE`, cues 0, 1, 2, 4, and 7 are coerced to 5, representing 'other' cues.
- **simplify_bino** A Boolean, with default `TRUE`, indicating whether or not sighting method codes should be simplified before being included as a covariate in detection function fitting. This can help to overcome the factor sample size limitations that may prevent inclusion as a covariate. If `TRUE`, methods other than 4 are all coerced to 5, representing 'other' methods.
- **detection_function_base:** The base key for the detection function, provided as a character vector. Accepted values are `"hn"` (half-normal key, the default, which exhibit greater stability when fitting to cetacean survey data; Gerrogette and Forcada 2005), `"hr"` (hazard-rate), or `c("hn", "hr")`, which will loop through both keys and attempt model fitting.
- **base_model:** The initial model formula, upon which to build using candidate covariates. if not provided by the user, the default is `"~ 1"`.
- **delta_aic:** The AICc difference between the model yielding the lowest AICc and other candidate models, used to define the best-fitting models. Typically, AICc differences of less than 2 (the default) indicate effectively equal model performance. If this value is not zero, then model averaging will be done: if multiple models are within `delta_aic` of the model with the lowest AICc, all "best" models will be used in subsequent steps and their results will be averaged. See Details below.

estimates

A nested list (i.e., a list of sublists), in which each constituent sublist contains the settings for a single density/abundance estimate. Check out the `LTabundR` function, `lta_estimates()`, which facilitates building this estimates argument. The slots below are recognized in each constituent sublist, but not all are required (i.e., some have defaults).

- **spp:** (Required) A character vector of species codes. If multiple codes

are provided, a single density/abundance estimate will be provided for this pooled group of species. If NULL, the codes in `fit_filters$spp` will be used.

- **title:** (Required) A title for this abundance estimate, given as a character vector, e.g., "Striped dolphin – pelagic". If left blank, the species code(s) will be concatenated to use as a title. Note that, if `spp_method` is 'each', then title must be the same length as `spp`.
- **years:** (Required) A required numeric vector of years, used to filter data to include only effort/sightings from these years.
- **regions:** (Required) A character vector of geostratum names, used to filter the data to a study area in which to estimate density/abundance. Any segment or sighting occurring within *any* of the provided regions will be returned. This holds true for nested regions: for example, in analyses from the Central North Pacific, in which the Hawaii EEZ geostratum ("HI_EEZ") is nested within the larger geostratum representing the entire CNP study area ("OtherCNP"), an input of `regions = "OtherCNP"` will return segments/sightings *both* inside the Hawaii EEZ *and* outside of it. These geostratum names must have been used to process this cruz object cohort during `LTabundR::process_surveys()`.
- **cruises:** (An optional numeric vector of cruise numbers, used to filter data to include effort/sighting from only certain cruises. Ignored if NULL.
- **regions_remove:** An optional character vector of geostratum names, similar to above. These regions will be subtracted from `regions` to determine the final study area in which density/abundance will be estimated. Any segment or sighting occurring within any of these `regions_remove` will **not** be used in density/abundance estimation. Using the example above, if `regions = "OtherCNP"` and `regions_remove = "HI_EEZ"`, only segments occurring within OtherCNP *and* outside of HI-EEZ will be used. This can be particularly useful for abundance estimates for pelagic stock that exclude nested insular stocks. These geostratum names must have been used to process this cruz object cohort during `LTabundR::process_surveys()`.
- **region_title:** An optional character vector indicating the title you would like to give to the region pertaining to this estimate. This can be useful if you have a complicated assemblage of regions you are combining and/or removing. If not supplied, the function will automatically generate a `region_title` based on `regions` and `regions_remove`.
- **g0:** (Optional) If left as the default NULL, this function will automatically estimate the weighted trackline detection probability ($g(0)$) according to the distribution of Beaufort sea states contained within the survey years/regions for which density/abundance is being estimated (this is done using the `LTabundR` function `g0_weighted()`; see its documentation for details). This will only be done if the `Rg0` input above is not NULL; if it is and you do not provide $g(0)$ values here, $g(0)$ will be coerced to equal 1. To coerce $g(0)$ to a certain value of your own choosing, you can provide a numeric vector of length 1 or 2. If length 1, this value represents $g(0)$ for all schools regardless of size. If length 2, these values represent $g(0)$ for small and large school sizes, as defined by `g0_threshold` below.

- `g0_cv`: (Optional) Similar to `g0` above: if left NULL, the CV of the $g(0)$ estimate will be automatically estimated based on weighted survey conditions. Alternatively, you can manually specify a CV here, using a numeric vector of length 1 or 2. If you do not specify a value and `Rg0` input is NULL, `g0_cv` will be coerced to equal 0.
- `g0_threshold`: (Optional) The school size threshold between small and large groups.
- `alt_g0_spp`: (Optional) An alternate species code to use to draw Relative $g(0)$ values from the `Rg0` input. This is useful in the event that $Rg(0)$ was not estimated for the species whose density/abundance you are estimating, but there is a similarly detectable species whose $Rg(0)$ parameters have been estimated.
- `combine_g0`: (Optional) A Boolean, with default FALSE. If TRUE, weighted $g0$ estimates will be produced *separately* for each species code provided (specifically, for each unique row in the `Rg0` table that is found after filtering by the species codes you provide in this estimate), *THEN* average those estimates together. This can be useful when you do not have a $Rg(0)$ estimates for a certain species, but you can approximate $Rg0$ by averaging together estimates from multiple species (e.g., averaging together weighted $g(0)$ from across roqual species in order to get a weighted $g(0)$ estimate for 'Unidentified roquals').
- `forced_effort` (Optional) If this is a single numeric value instead of NULL (NULL is the default), this value will be used as the survey effort, in km, in a brute-force method. If left NULL, the function will calculate survey effort itself. This is only helpful if you are looking for a relatively easy way to compare results from your own analysis to another (e.g., comparing LTabundR results to reports from NOAA reports prior to 2021, in which effort was calculated slightly differently).
- `area`: (Optional) If this is a single numeric value instead of NULL (NULL is the default), this value will be used as the area of the region in which abundance is being estimated, in square km, in a brute-force approach. If left NULL, the function will calculate the final area of the survey area resulting from the regions and `regions_remove` filters above.
- `remove_land`: (Optional) A Boolean, with default TRUE, indicating whether or not land area should be removed from the survey area before calculating its area for abundance estimation. This term is only referenced if `area` is not specified manually.

`use_g0` A Boolean, with default TRUE, indicating whether or not to use custom $g(0)$ value(s). If FALSE, the assumed $g(0)$ value will be 1. This can be a handy way of toggling weighted $g(0)$ estimation on and off across all sublists within estimates.

`abund_eff_types`

A character vector of EffType accepted as systematic effort (for density / abundance estimation). The default is just "S" (systematic effort), but in some surveys/cases you may wish to use fine-scale effort ("F") too.

`abund_bft_range`

A numeric vector of Beaufort Sea Sates accepted as systematic effort (for density / abundance estimation). The default is 0:6.

bootstraps	The number of bootstrap iterations. If 0 or 1, no bootstrapping will be carried out.
seed	Set a seed (any integer) to ensure that the bootstrap results are reproducible. If left NULL, the bootstrap results are liable to differ for each run of this function.
max_attempts	The maximum number of attempts for each bootstrap iteration; the bootstrap can fail at the detection-function fitting stage if the mrds df model fails to converge. This input sets the maximum number of tries before skipping the iteration and moving on, which means that, if max_attempts is reached for any iteration the total number of successful iterations in the final result may be less than the bootstraps input value.
results_file	If not NULL, this input will be taken as the name of the file in which to save the results as this function works. This can be a handy way of saving results as you go, in the event of a major error or system crash.
toplot	Boolean, with default TRUE, indicating whether detection function plots (<code>Distance::plot.ds()</code>) should be displayed as the candidate models are tested.
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Details

See the [vignette online](#) for detailed examples & case studies.

Survey area calculations: The area for which abundance is to be estimated is calculated separately for each sublist within estimates according to the inputs regions and regions_remove. This calculation is performed by a call to the LTabundR function `strata_area()`, which handles complex combinations and subtractions of geostrata, accounting for overlapping strata and the (optional) removal of any land area (see its documentation for details). The polygon for each resulting study area is added to the respective estimates sublist. Those polygons can be retrieved from the output's \$inputs slot.

Weighted $g(0)$ estimates: If $g(0)$ values are not supplied manually for an estimates sublist, a weighted $g(0)$ will be estimated as part of this function's operations through a call to the LTabundR function `g0_weighted()`, which automatically optimizes a model that estimates the $g(0)$ and its CV based on the distribution of effort in different Beaufort sea states within the specific year, region, and cruise in question. This is only done if the input `Rg0` is supplied.

Covariates in detection function estimation: Before detection functions are modelled, any covariates supplied by the user and specified as a factor are first tested for eligibility. Only factors with at least two levels (or whatever you specified with `df_settings$covariates_levels`) and 10 observations in each level (or whatever you specified with `df_settings$covariates_n_per_level`) are eligible for inclusion.

Fitting a detection function: The detection function is estimated using functions in the package mrds, primarily the main function `mrds::ddf()`, which uses a Horvitz-Thompson-like estimator to predict the probability of detection for each sighting. If multiple base key functions (e.g., half-normal or hazard-rate) are provided, and/or if covariates are specified, model fitting is done in a forward stepwise procedure: In the first round, the base model (no covariates, i.e., "`~1`") is fit first. In the second round, each covariate is added one at a time; at the end of the round, the covariate, if any, that produces the lowest AICc below the AICc from the previous round is added to the formula. This process is repeated in subsequent rounds, adding a new covariate term in each round, until the

AICc no longer improves. If a second base key is provided, the process is repeated for that second key. All models within `delta_aic` of the model with the lowest AICc qualify as best-fitting models.

The best-fitting model(s) is(are) then used to estimate the Effective Strip half-Width (ESW) based on the covariates associated with each sighting. If multiple best-fitting models occur, we will find the average ESW for each sighting across all models, using a weighted mean approach in which we weight according to model AICc. To turn off this model averaging step, set `delta_aic` to 0 to avoid passing multiple models to the abundance estimation stage.

This stage of the `lta()` command is executed within a backend function, `LTabundR::df_fit()`, which has its own documentation for your reference.

Note that if `LnSsTot` and/or `Bft` (or any other numeric covariate) are included as candidate covariates, missing data (in the case of `LnSsTot`, rows where `ss_valid == FALSE`), will be removed before detection function fitting, but those sightings will not be removed from the sightings data used for abundance estimation (see below). (Note that the code to handle these exceptions are contained within `lta()`, not `df_fit()`).

Estimating density & abundance: Estimates are produced for various combinations of species, regions, and years, according to the arguments specified in your `estimates` list(s). Before these estimates are produced, we filter the data used to fit the detection function to strictly systematic (design-based) effort (as specified in the `abund_eff_types` and `abund_bft_range` inputs). Note that if NA's occur in the `esw` column (due, for instance, to a covariate with missing data for a sighting), they will be replaced with the mean `esw` value for the remainder of the dataset in that region-year. Similarly, if `sightings` has a column named `ss_valid` (all standard `cruz` objects do) and any of the rows in that column are `FALSE`, those rows will have their best school size estimate (which will be NA or 1, since they are invalid) replaced by the mean best estimate for their respective species in the year for which abundance is being estimated. Currently the data used for that mean estimate are not specific to a given region, just the year of the abundance estimate.

This stage of the `lta()` command is executed within a back-end function, `LTabundR::abundance()`, which has its own documentation for your reference.

Bootstrap variance estimation: If the `bootstraps` input value is greater than 1, bootstrap variance estimation will be attempted. In each bootstrap iteration, survey segments are re-sampled with replacement before fitting the detection function and estimating density/abundance.

Note that the entire process is repeated in each bootstrap: step-wise fitting of the detection function, averaging of the best-fitting models, and density/abundance estimation for all species/region/year combinations specified in your `estimates` input. At the end of the bootstrap process, results are summarized for each species/region/year combination. 95% confidence intervals are calculated using the BCA method (adapted from package `coxed`, function `bca()`).

$g(0)$ values during bootstrapping: When conducting the non-parametric bootstrap routine to estimate the CV of density and abundance, uncertainty is incorporated into the $g(0)$ value in each iteration using a parametric bootstrapping subroutine: First, a logit-transformed distribution is modeled based upon the mean and CV of $g(0)$ provided by the user in the `estimates` input (see documentation for `LTabundR::g0_optimize()` for details on this step). This modeled distribution is used to randomly draw a $g(0)$ value for each iteration of the density/abundance bootstrap routine. In this way, the uncertainty in $g(0)$ is propagated into uncertainty in density/abundance.

Workflow recommendations: This function was designed to optimize workflow where possible. Some considerations:

- Expect a single `lta()` call for each species pool. You can use a single call to estimate the detection function once, then predict density/abundance separately for each species within the pool.
- Some inputs will be common across all species pools (e.g., `df_settings` and `bootstraps`). It may be most efficient – and easiest to keep consistent and to update – if you define these common inputs at the top of your script, then call them with a simple variable in each of your `lta()` calls.
- We recommend setting up each `lta()` call starting with `bootstraps = 0`, so that you can first test that the simple estimation step is successful for all species pools. Then test the bootstrapping functionality by setting `bootstraps` to 10 in each `lta()` call. Track how long it takes for your code to run, which you can use to predict processing time for a larger number of iterations, e.g., `bootstraps = 1000`.
- The most complicated argument to prepare is `estimates`. To help with this, `LTaundR` includes a function, `lta_estimates()`. See its documentation for details.

Value

A named list:

1. `pool`: The species pool this estimate pertains to.
2. `inputs`: A record of the inputs you provided, stored as a list.
3. `estimate`: A table of density/abundance estimates for each species/region/year combination specified in the `estimates` input. This data frame contains the following fields:
 - (a) `Region`: Name(s) of geostrata represented in this estimate.
 - (b) `Area`: Area of geostratum / region, in square km.
 - (c) `year`: Years represented in this estimate.
 - (d) `segments`: The number of effort segments used to estimate density/abundance.
 - (e) `km`: The km of trackline effort contained in these segments.
 - (f) `Area_covered`: The Area surveyed, according to km and `ESW_mean` (see next column).
 - (g) `ESW_mean`: Mean effective strip width, in kw, calculated as the mean probability of detection for all detections.
 - (h) `n`: The number of detections in the data.
 - (i) `g0_est`: The mean $g(0)$ estimate of detections, which may differ by group due to group size.
 - (j) `ER_clusters`: The encounter rate for detections (schools) (n / km)
 - (k) `D_clusters`: The density of detections (schools).
 - (l) `N_clusters`: The abundance of schools.
 - (m) `size_mean`: Average school size.
 - (n) `size_sd`: Standard deviation of school size.
 - (o) `ER`: Animal encounter rate.
 - (p) `D`: Animal density.
 - (q) `N`: Animal abundance.
 - (r) `g0_small`: the weighted $g(0)$ for small group sizes in this estimate.
 - (s) `g0_large`: the weighted $g(0)$ for large group sizes in this estimate.

- (t) `g0_cv_small`: the CV of weighted $g(0)$ for small group sizes in this estimate.
- (u) `g0_cv_large`: the CV of weighted $g(0)$ for large group sizes in this estimate.
- 4. `df`: A named list with details for the detection function.
 - (a) `best_models`: A `data.frame` summary of the best-fitting models, based upon the table produced by `Distance::summarize_ds_models()`. See that function's documentation for details.
 - (b) `all_models`: Similar to the preceding slot, a tabular summary of *all* models tested.
 - (c) `best_objects`: A list containing the `ds` objects (produced by package `Distance`) for each of the best-fitting models.
 - (d) `sample_size`: A `data.frame` with the detections for each species within the species pool used to fit the detection function (as well as Other species; see the `other_species` input). `Ntot` is total detections for each species; `Ndet` is total detections within the truncation distance and therefore used in the detection function fitting routine; `TD` is the truncation distance.
 - (e) `curve`: A `data.frame` of the best-fitting detection function curve (best-fitting models averaged together, weighted by AICc), for 100 distances between 0 and the `truncation_distance` (two columns: `km` and `p`, the probability of detection at that distance).
- 5. `g0_tables`: A list with $g(0)$ estimation parameters for each sublist in `estimates`.
- 6. `bootstrap`: A named list with results from the bootstrap process, only returned if the `bootstraps` input is greater than 1.
 - (a) `summary`: a `data.frame` with a row for each species/region/year combination for which density/abundance was estimated. Notable columns include `g0_mean` and `g0_cv` (the mean and CV of $g(0)$ values across parametric bootstrap iterations, which may differ slightly from the non-bootstrapped `g0_` estimates provided in the `estimate` slot above.); `Nmean` (the mean abundance, based on bootstrap re-sampling); `Nmedian` (median abundance); `Nsd` (standard-deviation of abundance); `CV` (coefficient of variation, which applies to both density and abundance); `L95` (the lower BCA 95% confidence interval), and `U95` (the upper BCA 95% confidence interval).
 - (b) `details`: a `data.frame` with details for every iteration of the bootstrap routine.
 - (c) `df`: a `data.frame` with the detection function curve for each bootstrap iteration.

lta_checks

Check for missing data before running lta().

Description

This function is designed for a QA/QC step before running the `lta()` function (line transect analysis). It checks for missing data in covariate columns, group size columns (if `lnsstot` is a desired covariate), and the perpendicular distance column (`PerpDistKm`). The function runs these checks twice: first for all sightings to be used in fitting the detection function (where missing data are not preferred but would not be fatal; sightings with incomplete data would simply be removed, lowering your sample size for the model fit), and second for detections that contribute to each point estimate (here missing data is disastrous; unless the missing data are filled in, the detection would be removed and the density estimate would be affected.) This function can be used to pinpoint and address holes in the data as the user sees fit.

Usage

```
lta_checks(cruz, df_settings, fit_filters, estimates)
```

Arguments

cruz	The same cruz object you will pass to lta(). See input details in that function's documentation.
df_settings	The same df_settings object you will pass to lta(). See input details in that function's documentation.
fit_filters	The same fit_filters object you will pass to lta(). See input details in that function's documentation.
estimates	The same estimates object you will pass to lta(). See input details in that function's documentation.

Value

The function prints messages to the Console as it performs its checks, and it also returns a list with all details needed to find & edit rows in the sightings data that have missing data.

lta_ci	<i>Compute confidence intervals of a bootstrapped LTA estimate</i>
--------	--

Description

Compute confidence intervals of a bootstrapped LTA estimate

Usage

```
lta_ci(estimate, bootstraps, ci = 0.95)
```

Arguments

estimate	The actual estimate
bootstraps	The bootstrapped values
ci	The percent confidence interval sought (as a decimal).

Value

A list with three versions of the 95% confidence interval, in each case returned as a two-element vector: \$quantile is the simple quantile-based CI of the bootstraps; \$bca is the bias-correction with acceleration; and \$bca_lognormal is the log-normal version of the bca CI. Adapted from Karin Forney's Excel magic.

lta_combine

Combine LTA estimates from separate regions

Description

Combine the results of `LTabundR::lta()` from separate regions, weighting density by respective area and handling the combination of CV and 95% confidence interval estimates.

Usage

```
lta_combine(d1, n1, cv1, lci1, uci1, area1, d2, n2, cv2, lci2, uci2, area2)
```

Arguments

d1	Density estimate (animals per square km) from region 1.
n1	Abundance estimate from region 1.
cv1	CV estimate from region 1.
lci1	Lower 95% confidence interval of abundance from region 1.
uci1	Upper 95% confidence interval of abundance from region 1.
area1	Area, in square km, of region 1.
d2	Density estimate for region 2.
n2	Abundance estimate for region 2.
cv2	CV estimate for region 2.
lci2	LCI for region 2.
uci2	UCI for region 2.
area2	Area for region 2.

Value

A one-row data.frame with these columns:

1. D: the combined density (animals per square km), weighted by the areas of constituent regions.
2. N: the combined abundance.
3. CV: the combined CV of density and abundance.
4. LCI: the lower 95% confidence interval of abundance.
5. UCI: the upper 95% confidence interval of abundance.
6. area: combined area.

lta_delta_method	<i>Estimate CV and 95% confidence intervals with the delta method</i>
------------------	---

Description

This function loops through a list of results from LTabundR: :lta() and updates their CV and 95% CI estimates, which were produced using a bootstrap method, using new estimates based on the delta method.

Usage

```
lta_delta_method(ltas, verbose = TRUE)
```

Arguments

ltas	A list of outputs from the function lta().
verbose	Boolean; print report of new estimates to the console?

Value

An updated version of the ltas input with new values for CV, L95, and U95 in the \$bootstrap\$summary slot.

lta_destratify	<i>Pool stratified LTA results</i>
----------------	------------------------------------

Description

Loop through a list of stratified line-transect-analysis results and "destratify" – i.e., pool – them into a single estimate.

Usage

```
lta_destratify(
  lta_list,
  years,
  combine_method = "arithmetic",
  new_region = NULL,
  verbose = TRUE
)
```

Arguments

<code>lta_list</code>	A list of line-transect-analysis results produced from <code>LTabundR::lta()</code> and compiled into a list with <code>LTabundR::lta_enlist()</code> .
<code>years</code>	A numeric vector of year(s) in which you want to pool all region-stratified results into a single annual estimate.
<code>combine_method</code>	A character string indicating the means by which estimates of the CV and 95% confidence interval of abundance will be destratified. The two recognized options are "arithmetic" (the default) and "bootstrap". See details below.
<code>new_region</code>	A name for the pooled geostratum represented by the pooled estimate. If this is not provided, a name will be generated from the geostrata that are pooled.
<code>verbose</code>	A Boolean, with default TRUE, indicating whether or not status updates should be printed to the Console.

Details

Stratified estimates of parameter means are pooled using a weighted mean in which the area of each geostratum is used as the weight. Stratified estimates of abundance variance terms (SD, CV, and 95% confidence interval) are pooled either arithmetically – using standard equations to pool estimates one at a time – or iteratively – using the bootstrap results contained within the `lta_list`. The iterative technique resamples the abundance bootstrap estimates from each geostratum 10,000 times then calculates SD, CV, and 95% confidence interval from the resulting distribution. Some variance terms, e.g., the SD of school size and the CV of $g(0)$, are not destratified in the current version and are instead replaced with NA.

Value

A modified `lta_list`, in which stratified estimates have been pooled into single annual estimates for relevant years.

<code>lta_diagnostics</code>	<i>Run diagnostics on a LTA result</i>
------------------------------	--

Description

Run diagnostics on a LTA result

Usage

```
lta_diagnostics(
  lta_result,
  options = 1:8,
  describe_options = FALSE,
  wait = TRUE
)
```

Arguments

lta_result	The result of lta() for a single species pool.
options	Numeric vector indicating which output options to return. These options are printed for the user when describe_options is TRUE.
describe_options	Boolean; if TRUE, the function will print a list of output options for the user. This helps the user determine which numbers to request with the input options.
wait	Boolean; if TRUE, the function will wait for the user to press <Enter> in between each output option.

Value

A series of printed tables and plots, which the user can step through using the Enter key.

lta_enlist	<i>Compile a directory of LTA results into a list</i>
------------	---

Description

This function reads in the line-transect-analysis results (from LTabundR: :lta()) within a directory and concatenates them into a list.

Usage

```
lta_enlist(lta_path)
```

Arguments

lta_path	The path to a directory containing .RData files, each containing a object resulting from LTabundR: :lta(). Typically, each RData file contains the product of lta() for a single species or species pool. See the vignette case studies for examples.
----------	---

Value

A list of LTA results.

lta_estimates

Build a function that creates estimates sub-lists for lta()

Description

This function can be used to efficiently build up sub-lists for the estimates input that you must pass to the lta() function for line-transect analysis. Most studies of multiple species endeavor to estimate density/abundance for each species in the same set of year-region scenarios. Doing so typically requires multiple estimates sub-lists for each species of interest. Writing those sub-lists manually can be redundant, tedious, and prone to inconsistencies or error. To ensure the exact same scenarios are applied to each species of interest, you can use this function to return a custom function that you can then use to create a standard set of sub-lists for each of your species.

Usage

```
lta_estimates(scenarios)
```

Arguments

scenarios A list of sub-lists. Each sublist specifies one scenario for a density/abundance estimate. The accepted names within each sublist are:

- years
- cruises
- regions
- regions_remove
- region_title
- forced_effort
- area
- remove_land

These arguments are explained under the estimates input for the lta() function.

Value

This function returns a custom function named estimator(), which accepts the following inputs:

- spp (required)
- title (required)
- g0 (the remainder are optional)
- g0_cv
- g0_threshold
- alt_g0_spp
- combine_g0

These arguments are explained under the `estimates` input for the `lta()` function. The `estimator()` function will return a list of sub-lists, in which the species-specific inputs it accepts are added to each sub-list in the `scenarios` argument you provided above. The result of the `estimator()` function is what you can pass as the `estimates` argument in `lta()`. See the [vignette case studies](#) for examples.

`lta_plot`*Plot LTA results*

Description

This function returns abundance estimate plots (i.e., with year on the x axis, abundance on the y axis, the best estimate represented by a point, and the confidence interval represented by a vertical line). This function is capable of producing multi-panel plots for multiple species, and it is designed to produce a dynamic layout according to the number of species pools provided.

Usage

```
lta_plot(lta_result, species = NULL, years = NULL, nrows = NULL, ncols = NULL)
```

Arguments

<code>lta_result</code>	The result of <code>lta()</code> or <code>lta_enlist()</code> . See their documentation for details.
<code>species</code>	A character vector of estimate titles. If <code>NULL</code> , all titles within <code>lta_result</code> will be used.
<code>years</code>	A numeric vector of years whose estimates will be included in all plots, even if an estimate is not available for a given species (to make the x-ranges of all plots equal)
<code>nrows</code>	Number of rows in multi-panel plot. Can be <code>NULL</code> (which is the default).
<code>ncols</code>	Number of columns in multi-panel plot. Can be <code>NULL</code> (which is the default).

Value

A `ggplot2` plot, or many `ggplot2` plots arranged by `gridExtra::grid.arrange()`.

lta_pool	<i>Pool bootstraps from multiple lta() outputs</i>
----------	--

Description

This function allows you to pool, or concatenate, the results of two separate `lta()` runs in order to achieve the desired number of bootstrap iterations. An example use case: you are running a 1,000-iteration `lta()` call overnight, but the power goes out at iteration 900. In the morning you can run a 100-iteration version of the same `lta()` call, then use this function to pool the two results together.

Usage

```
lta_pool(ltas, bootstraps = NULL)
```

Arguments

<code>ltas</code>	A list of outputs from the function <code>lta()</code> .
<code>bootstraps</code>	Desired number of bootstraps to keep. Example use case: You have two <code>lta()</code> outputs, one with 600 iterations and another with 500. You want to pool the two outputs but only keep 1,000 iterations total. Specify <code>bootstraps = 1000</code> and this function will randomly select 1,000 of the 1,100 bootstraps available.

Value

An `lta()` output object. The `$estimate` slot will be the exact same as the first slot in the `ltas` input; The `$details` slot will have the pooled bootstrap samples. The `$df` slot will have the pooled detection function curves. The `$summary` slot will have updated summary statistics based on the pooled set of bootstraps.

lta_report	<i>Build LTA tables for standard reports</i>
------------	--

Description

This function formats the results from an LTA analysis (`LTabundR::lta()`) into that expected for standard tables in NOAA stock assessment reports (Tables 1, 2, 3, and 4, as well as appendix tables).

Usage

```
lta_report(lta_result, cruz = NULL, verbose = TRUE)
```

Arguments

lta_result	An object holding the result of <code>LTabundR::lta()</code> or <code>LTabundR::lta_enlist()</code> .
cruz	The cruz object (produced from <code>LTabundR::process_surveys()</code>) that was passed to <code>lta()</code> to produce <code>lta_result</code> . This is optional; if not supplied, only part of Table 1 will be able to be filled in and <code>tableA2</code> will not be provided (see Value below).
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console.

Value

A list with five slots:

1. `table1a`: Sighting counts for all species in the cruz dataset for the years in which estimates were prepared. If the cruz argument was NULL, this slot is also NULL. If not, a count of all sightings and systematic-only sightings (i.e., `EffType = "S"` and `Bft <= 6`) are given for each species-year-region in the cruz data. These counts are provided separately from the `table1b` slot below, since those counts are based on the `lta_result` object, and will not include sightings for species that did not have a specific LTA estimate specified when it was made. We also include this separately so as to give the user full flexibility in how they summarize sighting counts by region/population/stock.
2. `table1b`: Sighting counts used in estimates of density/abundance. Columns are prepared for total sightings (`Ntot`) and systematic sightings (`Nsys`), but they are left blank, since it is not clear how sightings from multiple regions in `table1a` would be concatenated for this table, since that involves stock-level designations. The user can fill in those gaps accordingly.
3. `table2`: Sample sizes and truncation distances for detection functions modeled by using pooled species/sightings
4. `table3`: Line-transect parameter estimates (mean Effective Strip half-Width, `ESW`; mean school size; trackline detection probability, `g(0)`; and its CV).
5. `table4`: Density and abundance estimates.
6. `tableA1`: Appendix table with study areas for each geostratum/cohort/year.
7. `tableA2`: A list in which each slot is a `data.frame` with Effort & Beaufort allocation, parsed by geostratum, for each year. If the cruz argument was NULL, this slot is also NULL.

lta_result

Example of LTA results for three delphinids in 2010 and 2017

Description

Example of LTA results for three delphinids in 2010 and 2017

Usage

```
lta_result
```

Format

An object of class `list` of length 6.

Details

This dataset was processed using the following code:

```
data("cnp_150km_1986_2020")
cruz <- cnp_150km_1986_2020
cruz$strata

data('g0_results')
Rg0 <- g0_results

fit_filters = list(spp = c('013', '026', '031'), # striped, frasers, melon-headed
                  pool = 'Multi-species pool 1',
                  cohort = 'all',
                  truncation_distance = 5,
                  other_species = 'remove',
                  years = 1986:2017,
                  regions = NULL,
                  not_regions = NULL)

df_settings = list(covariates = c('bft', 'lnsstot', 'cruise', 'year', 'ship', 'species'),
                  covariates_factor = c(FALSE, FALSE, TRUE, TRUE, TRUE, TRUE),
                  covariates_levels = 2,
                  covariates_n_per_level = 10,
                  detection_function_base = 'hn',
                  base_model = '~1',
                  delta_aic = 2)

estimates <-
list(
  list(spp = '013',
       title = 'Striped dolphin',
       years = 2010,
       regions = 'HI_EEZ',
       g0 = 0.33, g0_cv = 0.20),
  list(spp = '013',
       title = 'Striped dolphin',
       years = 2017,
       regions = 'HI_EEZ',
       g0 = 0.32, g0_cv = 0.21),
  list(spp = '026',
       title = "Fraser's dolphin",
       years = 2010,
       regions = 'HI_EEZ',
       g0 = 0.33, g0_cv = 0.20),
```

```

list(spp = '026',
     title = "Fraser's dolphin",
     years = 2017,
     regions = 'HI_EEZ',
     g0 = 0.32, g0_cv = 0.21),
list(spp = '031',
     title = 'Melon-headed whale',
     years = 2010,
     regions = 'HI_EEZ',
     g0 = 0.33, g0_cv = 0.20),
list(spp = '031',
     title = 'Melon-headed whale',
     years = 2017,
     regions = 'HI_EEZ',
     g0 = 0.32, g0_cv = 0.21))

result <- lta(cruz,
             Rg0,
             fit_filters,
             df_settings,
             estimates,
             use_g0 = TRUE,
             bootstraps = 100,
             toplot=TRUE,
             verbose=TRUE)

```

lta_subgroup

Subgroup-based line transect analysis

Description

A flexible routine for carrying out subgroup-based line-transect analysis using the methodologies described for Hawai'ian false killer whales, *Pseudorca crassidens*, in [Bradford et al. \(2020\)](#). The function returns an estimate of density and abundance – along with estimates of intermediate parameters – with a CV derived from a bootstrapping routine. As part of this process, relative track-line detection probability ($g(0)$) is modeled as a function of Beaufort sea state (using LTabndR function `g0_model()`), then a weighted $g(0)$ and its CV are estimated using LTabundR function `g0_weighted()`.

Usage

```

lta_subgroup(
  df_sits,
  truncation_distance,
  ss,
  density_segments,
  density_das,

```

```

density_sightings,
df_settings = list(covariates = NULL, covariates_factor = NULL, covariates_levels = 2,
  covariates_n_per_level = 10, simplify_cue = TRUE, simplify_bino = TRUE,
  detection_function_base = "hn", base_model = "~1", delta_aic = 2),
Rg0 = NULL,
cruz10 = NULL,
g0_spp = NULL,
g0_truncation = NULL,
g0_constrain_shape = FALSE,
g0_jackknife_fraction = 0.1,
abundance_area = NULL,
iterations = 5000,
seed = NULL,
output_dir = NULL,
toplot = FALSE,
verbose = TRUE
)

```

Arguments

<code>df_sits</code>	(Required.) A data.frame of sightings you want to use to fit the detection function model. For false killer whales in Bradford et al. (2020), this is a combination of systematic sightings prior to 2010 and Phase 1 sightings from 2010 onwards (using the PC protocol). This dataframe must have a column named <code>PerpDistKM</code> with detection distances in km. No filtering will be applied to these sightings within this function, so make sure you provide the data pre-filtered. Bradford et al. (2020) used a single detection function for all populations of false killer whale. Note that all column names within <code>df_sits</code> must be within <code>density_sightings</code> and the names must match exactly. This is needed in order to predict the Effective Strip Width of <code>density_sightings</code> based on the detection function fitting procedure.
<code>truncation_distance</code>	(Required.) The truncation distance, in km, to apply during detection function model fitting.
<code>ss</code>	(Required.) A numeric vector of subgroup school sizes to use to find its mean and bootstrapped CV. In Bradford et al. (2020), data come from all Phase 1 and Phase 2 estimates of subgroup sizes from 2010 onwards. These estimates are the geometric mean of repeat estimates from separate observers.
<code>density_segments</code>	(Required.) The survey segments to be used in density/abundance estimation. For example, Bradford et al. (2020) used 150-km segments to estimate false killer whale density in the Hawaiian EEZ in 2017 (these data are available in the built-in dataset <code>"cnp_150km_1986_2020"</code>). No filtering will be applied to these segments, so make sure only the segments you wish to use are included and nothing more. For example, in the case above, make sure you are only providing systematic segments for the Hawaiian EEZ in 2017.
<code>density_das</code>	(Required.) The complete survey data corresponding to the above segments. These data will be used to determine the proportion of survey effort occurring

	in each Beaufort sea state during Relative $g(0)$ estimation.
density_sightings	(Required.) The encounters to use in density/abundance estimation. In Bradford et al. (2020), these were the Phase 1 detections of false killer whales within the population-region-year of interest, e.g., Northwest Hawaiian Island population sightings within the Hawaiian EEZ in 2017. No filtering will be applied to these sightings, so make sure only the sightings you wish to use are included and nothing more. Note that all column names within <code>df_sits</code> must be within <code>density_sightings</code> and the names must match exactly. This is needed in order to predict the Effective Strip Width of <code>density_sightings</code> based on the detection function fitting procedure.
df_settings	Optional. A named list, with parameters for fitting the detection function. See detailed documentation in the <code>lta()</code> function. Default inputs shown above. Note that the default is that covariates are <i>not</i> used in detection function fitting.
Rg0	A <code>data.frame</code> with estimates of Relative $g(0)$ and its CV at each Bft state. If this input is left NULL, then these estimates will be produced by the function using the subsequent <code>g0_</code> inputs. If this input is not supplied and any of the subsequent <code>g0_</code> inputs are missing, then $g(0)$ will be assumed to be 1.0 with CV of 0.0. If supplied, this <code>data.frame</code> has three required columns: <code>bft</code> (Beaufort sea state, numbers between 0 and 7), <code>Rg0</code> ($Rg(0)$ estimates for each Beaufort state), and <code>Rg0_CV</code> (the CV of the $Rg(0)$ estimate in each Beaufort state). Other columns are allowed but will be ignored.
cruz10	A processed <code>cruz</code> object with short segment lengths, ideally 10 km or less (hence the 10 in the input name). This <code>cruz</code> object will be used to estimate $Rg(0)$, i.e., the relative trackline detection probability. Consider using the built-in dataset "noaa_10km_1986_2020".
g0_spp	A character vector of species codes to use to estimate $Rg(0)$. In most cases this will be a single species, e.g., '033' for false killer whales. Not required if the <code>Rg0</code> input is supplied.
g0_truncation	The truncation distance to use when estimating $Rg(0)$. In Bradford et al. (2020) this is 5.5 km.
g0_constrain_shape	Some $Rg(0)$ curves will not decline monotonically due to sample size issues at low Bft (0-2) or high Bft (5-6) states. To coerce monotonic decline, set this input to TRUE, and the function will use a shape-constrained GAM (<code>scam()</code> from package <code>scam</code>) instead of a classic <code>mgcv::gam()</code> .
g0_jackknife_fraction	The proportion of data to leave out within each jackknife permutation for estimating the CV of $g(0)$. The default is 0.1 (i.e., 10% of the data, yielding 10 jackknife loops), after Barlow (2015).
abundance_area	The area in square km of the region of interest. The density estimate will be scaled by this area.
iterations	Number of iterations to use in the various CV bootstrapping procedures occurring throughout this function, specifically: Effective Strip Half-Width CV estimation, school size CV estimation, weighted $g(0)$ CV estimation, encounter rate estimation, and density/abundance estimation.

seed	Set a seed (any integer) to ensure that the bootstrap results are reproducible. If left NULL, the bootstrap results are liable to differ slightly for each run of this function.
output_dir	The path in which results RData files should be stored. If left NULL, no results will be stored. To use your current working directory, simply provide "".
toplot	A Boolean, with default FALSE, indicating whether to plot various aspects of the analysis.
verbose	A Boolean, with default TRUE, indicating whether to print status updates to the Console.

Details

This function performs the following operations:

1. Fits a detection function to `df_sits`, using the LTabundR function `df_fit()`. That function will be used to estimate the effective strip half-width (ESW) of each sighting in `detection_sightings` (and bootstrapped versions thereof).
2. Conducts bootstrap re-sampling of the detection function fitting routine in order to estimate the CV of ESW.
3. Estimates the arithmetic mean of subgroup school size based on the `ss` input.
4. Creates a bootstrap-resampled distribution of subgroup school sizes, with which CV is estimated.
5. Models the relative $g(0)$ in different survey conditions using the LTabundR function `g0_model()`. This function also estimates the CV of the $Rg(0)$ estimate in each Beaufort sea state using jackknife resampling.
6. Estimates the ESW and encounter rate (subgroup detections / trackline surveyed).
7. Creates a bootstrap-resampled distribution of encounter rate and ESW estimates.
8. Calculates a weighted $g(0)$ estimate according to the proportion of effort occurring in each Beaufort sea state, then uses an automated parameter optimization routine (see details in LTabundR function `g0_weighted()`) to estimate the CV of the weighted $g(0)$ estimate.
9. Creates a bootstrap-resampled distribution of the weighted $g(0)$ estimate.
10. Estimates density using the best estimates of effective strip half-width, school size, $g(0)$, and the encounter rate.
11. Estimates abundance by scaling the density estimate by the provided `abundance_area`, if provided.
12. Creates a bootstrap-resampled distribution of the density estimate by iteratively drawing values (without replacement, since the constituent distributions were already built with replacement) from the resampled distributions of the constituent parameters of the density equation.
13. Creates a bootstrap-resampled distribution of the abundance estimate by scaling the density distribution by `abundance_area`.

Value

A list with these slots: `$estimate` (point estimates of parameters, detailed below), `$bft` (proportion of survey effort in each sea state), `$g0_details` (details on g_0 fitting), `$df` (details on detection function fitting), `$bootstraps` (parameter values for each bootstrap iteration), `$iterations` (the number of iterations). Columns in `$estimate` slot are as follows:

1. D: The estimate of density for the population.
2. D_CV: The CV of the density estimate.
3. D_L95: The lower 95% confidence interval of density using the BCA method.
4. D_U95: The upper 95% confidence interval of density using the BCA method.
5. N: The estimate of abundance.
6. N_CV: The CV of the abundance estimate.
7. N_L95: The lower 95% confidence interval of abundance using the BCA method.
8. N_U95: The upper 95% confidence interval of abundance using the BCA method.
9. ER: The estimate of the encounter rate.
10. ESW = The estimate of the Effective Strip Width (km) for the population.
11. ESW_CV = Estimate of ESW CV, based on standard deviation of bootstrap estimates of ESW.
12. ss = Mean subgroup size estimate.
13. n: Number of sightings used in density estimation.
14. L: Survey effort, in km, used in density estimation.
15. n_segments: Number of effort segments used in density estimation.
16. g_0 : The empirical weighted mean of $g(0)$ for the point estimate, based on sightings conditions in `density_segments`.
17. g_0_{cv} : The CV of this estimate of the point estimate of the weighted mean of $g(0)$, as estimated by an MCMC routine.
18. $g_0_details$: A list with detailed results from $Rg(0)$ estimation (see output details in `?g0_model`).
19. `df`: A list with detailed results from detection function estimation (see output details in `?df_fit`).
20. `bootstraps`: A named list with the bootstrapped values for `esw` (effective strip half-width), `ss` (subgroup size), `g0` (relative $g(0)$), `er` (encounter rate), `D` (density), and `N` (abundance).
21. `iterations`: number of bootstrap iterations used for CV estimation.

See the [online vignette](#) for more details

map_cruz

*Interactive map for Wincruz survey***Description**

Produced an interactive leaflet map of your Wincruz survey data.

Usage

```
map_cruz(
  cruz,
  cohort = 1,
  eez_show = TRUE,
  eez_color = "black",
  eez_weight = 0.6,
  eez_opacity = 0.5,
  strata_show = TRUE,
  strata_color = "forestgreen",
  strata_weight = 1,
  strata_opacity = 0.5,
  effort_show = FALSE,
  effort_color = "darkblue",
  effort_weight = 0.6,
  effort_opacity = 0.85,
  sightings_show = TRUE,
  sightings_color = "color code",
  sightings_radius = 1,
  sightings_opacity = 0.5,
  initial_zoom = 4,
  verbose = TRUE
)
```

Arguments

cruz	Your cruz object (produced from LTabundR::process_surveys()).
cohort	The cohort whose data you would like to map, provided as a number indicating which slot in cruz\$cohorts should be referenced.
eez_show	Boolean; display EEZ boundaries?
eez_color	Character string; color of EEZ boundary lines, if eez_show is TRUE.
eez_weight	Single positive numeric; weight (thickness) of EEZ boundary lines, if eez_show is TRUE.
eez_opacity	Single numeric between 0 and 1; opacity of EEZ boundary lines, if eez_show is TRUE.
strata_show	Boolean; display geostratum boundaries?
strata_color	Character string; color of geostratum boundary lines, if strata_show is TRUE.

strata_weight	Single positive numeric; weight (thickness) of geostratum boundary lines, if strata_show is TRUE.
strata_opacity	Single numeric between 0 and 1; opacity of geostratum boundary lines, if strata_show is TRUE.
effort_show	Boolean; display survey tracks? Default is FALSE, because rendering the track-lines can take a while. If verbose is TRUE, a progress bar will be printed to the console as the survey tracklines are rendered.
effort_color	Single character string; color of survey tracklines, if effort_show is TRUE.
effort_weight	Single positive numeric; weight (thickness) of survey tracklines, if effort_show is TRUE.
effort_opacity	Single numeric between 0 and 1; opacity of survey tracklines, if effort_show is TRUE.
sightings_show	Boolean; display sightings?
sightings_color	A single character string, indicating the color for sighting markers. Note: if you enter "color code", the sightings will be color-coded by leaflet according to species code, and a legend will be provided; this argument is ignored if sightings_show is FALSE.
sightings_radius	A single positive numeric, indicating the size of the sightings markers; this argument is ignored if sightings_show is FALSE.
sightings_opacity	A single numeric between 0 and 1, indicating the opacity of sightings markers; this argument is ignored if sightings_show is FALSE.
initial_zoom	Initial zoom default for the leaflet map.
verbose	Boolean, with default TRUE, indicating whether or not updates should be printed to the Console. (The only updates occur in the preparation of effort tracks, if effort_show is TRUE)

Value

An interactive leaflet map.

mixed_group_affiliates

Find affiliated species from mixed-species groups

Description

Find affiliated species from mixed-species groups

Usage

```
mixed_group_affiliates(cruz, cohort = 1, spp)
```

Arguments

cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).
cohort	The cohort whose sightings you would like to recalibrate, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.
spp	Character vector of species codes. Groups containing these species will be found, and all other species present in their mixed-species groups will be returned as "affiliates".

Value

A character vector of species codes for all species affiliated with `spp` in mixed-species groups, provided in increasing order of frequency.

noaa_10km_1986_2020	<i>Cruise data processed for all NOAA SWFSC and PIFSC surveys, 1986 - 2020 (10 km segment lengths)</i>
---------------------	--

Description

Cruise data processed for all NOAA SWFSC and PIFSC surveys, 1986 - 2020 (10 km segment lengths)

Usage

```
noaa_10km_1986_2020
```

Format

An object of class `list` of length 3.

Details

This dataset was processed using the following code:

```
data(group_size_coefficients)

survey <- load_survey_settings(
  out_handling = 'stratum',
  max_row_interval = Inf,
  segment_method = "equallength",
  segment_target_km = 10,
  segment_max_interval = 6,
  segment_remainder_handling = c("segment"),
  group_size_coefficients = group_size_coefficients)

data(strata_ccs)
```

```

data(strata_cnp)
strata <- c(strata_ccs, strata_cnp)
strata$NEPAC <- data.frame(Lat = c(-20, -20, 65, 65), Lon = c(-230, -70, -70, -200))

all_species <- load_cohort_settings(
  id = "all",
  species = NULL,
  probable_species = FALSE,
  sighting_method = 0,
  cue_range = 0:7,
  school_size_range = c(0, 10000),
  school_size_calibrate = TRUE,
  calibration_floor = 0,
  use_low_if_na = TRUE,
  io_sightings = 0,
  geometric_mean_group = TRUE,
  truncation_km = 7.5,
  beaufort_range = 0:6,
  abeam_sightings = FALSE,
  strata_overlap_handling = c("smallest"),
  distance_types = c('S', 'F', 'N'),
  distance_modes = c('P', 'C'),
  distance_on_off = TRUE
)

settings <- load_settings(strata = strata,
  study_area = NULL,
  survey = survey,
  cohorts = list(all_species))

das_file = c('../test_code/data/swfsc_1986_2020.das')
swfsc <- process_surveys(das_file,
  settings = settings)

das_file = '../test_code/CNP/CenPac1986-2020_Final_alb.das'
pifsc <- process_surveys(das_file,
  settings = settings)

cruzes <- list(swfsc, pifsc)
noaa_10km_1986_2020 <- cruz_combine(cruzes)

```

points_to_line

Convert coordinates to a sf line

Description

This is an internal function typically not called by a user directly.

Usage

```
points_to_line(data, long, lat, id_field = NULL, sort_field = NULL)
```

Arguments

data	The dataframe containing columns with coordinates.
long	A character vector with the name of the column with longitude coordinates.
lat	A character vector with the name of the column with latitude coordinates.
id_field	If the dataframe contains data for multiple lines, specify the name of the column (as a character vector) that indicates which line each point belongs to.
sort_field	If the dataframe is not arranged in the sequence you wish, specify the name of the column (as a character vector) you want to arrange by before creating the line.

Value

A `sp::SpatialLines` object created from `sp::spRbind()`.

```
prep_bootstrap_datasets
```

Prepare bootstrapped versions of datasets

Description

An interior function, typically not called by analysts. This function returns segment and sighting data according to resampled segment IDs, which serve as part of the bootstrap variance estimation process. The function includes an option to specify a selection of segment IDs, which allows the same bootstrap segments to be selected for the abundance estimation stage as was used in the detection function model fitting stage.

Usage

```
prep_bootstrap_datasets(segments, sightings, segment_picks = NULL, seed = NULL)
```

Arguments

segments	Effort segments dataframe, drawn from a <code>cruz</code> object (e.g., <code>cruz\$cohorts[[1]]\$segments</code>), already filtered to contain the effort you wish to use to fit the detection function.
sightings	Sightings dataframe, drawn from a <code>cruz</code> object (e.g., <code>cruz\$cohorts[[1]]\$sightings</code>), already filtered to contain the sightings you wish to use to fit the detection function.
segment_picks	If segment IDs have been previously selected for this dataset, provide them here.
seed	Set a seed (any integer) to ensure that the result is reproducible. If left <code>NULL</code> , the results are liable to differ for each run of this function.

Value

A list with resampled data: segments, sightings, and segment_picks (a numeric vector of segment IDs that have been re-sampled).

process_km	<i>Calculate distances between rows in a DAS file</i>
------------	---

Description

This function is the near-equivalent of the DISTRAP subroutine in ABUND9. This is an internal function typically not called by a user directly. It is called in format_das().

Usage

```
process_km(
  das,
  min_interval = 5,
  max_interval = 3600,
  replacement_interval = 3600,
  max_km_gap = 100,
  km_filler = 1,
  speed_filler = 10 * 1.852,
  debug_mode = TRUE
)
```

Arguments

das	A dataframe of DAS survey data.
min_interval	Minimum seconds between rows before distance will be calculated. Default is 30 seconds; if the interval is less than this value, the distance will be assumed to be 0 km.
max_interval	The maximum time interval, in seconds, between rows before assuming that there has been a break in survey data logging. This time interval will be replaced with the value specified in replacement_interval.
replacement_interval	The time interval, in second, to use as a replacement when the time interval between rows exceeds max_interval. The default is 900 seconds. or the max_interval specified, whichever is smaller.
max_km_gap	Another way of avoiding long gaps in data; this is the maximum distance gap, in km, allowed between rows of data. This constraint is applied <i>after</i> the interval constraints above.
km_filler	When valid speed and position information is not available (e.g., the given distance exceeds max_km_gap), this value (in km) will be used as an estimate of the distance in between consecutive rows of data.

speed_filler	When speed is not available in the data, this value (in kmh) will be used as a filler in order to estimate the distance between consecutive rows of data based on timestamp differences (when lat/long coordinates are not available).
debug_mode	Boolean, with default FALSE, indicating whether details should be printed to the Console that facilitate debugging.

Value

A numeric vector, the same length as the number of rows in `das`, of distances (in km). The final element of this vector will be 0.

process_polygon	<i>Format a CSV into a polygon for mapping</i>
-----------------	--

Description

This is an internal function typically not called by a user directly. It takes a dataframe of polygon coordinates and returns a list with the polygon formatted in various ways, as well as a calculation of the area of the polygon *with land area removed* (optionally).

Usage

```
process_polygon(polygon_dataframe, remove_land = TRUE, toplot = FALSE)
```

Arguments

polygon_dataframe	A dataframe in which the first two columns are Lon and Lat, providing coordinates in decimal degrees in which South and West coordinates are negative. It is acceptable if vertices in the eastern hemisphere are described using negative longitudes below -180, e.g., -185. (The function will correct these to proper decimal degrees, e.g., -185 will become 175.) Other columns are allowed, but the first two need to be Lon and Lat (in that order).
remove_land	If TRUE, the area of land occurring within the polygon will be subtracted from the polygon's area, so that the reported area represents aquatic habitat only.
toplot	If TRUE (not the default), a map of the polygon will be plotted, and – if <code>remove_land</code> is TRUE – any land inside will be plotted in grey.

Value

A list with four named slots:

1. `coords` The input dataframe, with longitudes corrected to proper negative decimal degrees, e.g., longitudes supplied as -185 will become 175.
2. `polygon` The polygon created by `sf::st_polygon()`
3. `sf` The `sf` version of the polygon that will be passed to mapping functions.

4. `sf_land` The `sf` version of the land polygons included within the polygon, if any.
5. `km2` The area of the polygon, in square kilometers.
6. `idl` A Boolean indicating whether or not this polygon crosses the International Date Line.
7. `lat_range` Latitude range of polygon, in decimal degrees (N positive, S negative).
8. `lon_range` Longitude range of polygon, in decimal degrees (E positive, W negative).
9. `plot` A `ggplot` object; NULL if the input `toplot` is FALSE.

<code>process_sightings</code>	<i>Process survey sightings</i>
--------------------------------	---------------------------------

Description

This is an internal function typically not called by a user directly. It is the fifth subroutine called within `process_surveys()`, after `segmentize()` and before `process_subgroups()`.

Usage

```
process_sightings(
  cruz,
  calibrate = NULL,
  calibrate_floor = NULL,
  geometric_mean = NULL,
  verbose = TRUE
)
```

Arguments

<code>cruz</code>	A <code>cruz</code> object passed from <code>segmentize()</code> .
<code>calibrate</code>	An argument allowing you to override the settings contained within the <code>cruz</code> object. If NULL, the <code>cruz</code> settings will be used. If NA, calibration will be skipped. Otherwise supply a list to specify which calibration approach to use. Currently two approaches are supported: that used by ABUND 7/8 (the default shown), which takes an observer-specific, species-agnostic approach, and that developed in Gerrodette et al. (2019) and applied in Barlow et al. (2026), which includes a species-specific, observer-agnostic approach. Details in <code>?load_cohort_settings()</code> .
<code>calibrate_floor</code>	Another argument allowing for settings override. This argument accepts a number indicating the minimum raw school size estimate for which school size calibration will be attempted.
<code>geometric_mean</code>	Another argument allowing settings override. This argument accepts a Boolean; if TRUE, geometric means will be calculated instead of arithmetic means. If school size calibration is carried out, the geometric mean will be weighted by calibration variance, such that estimates from observers with low variance will receive more weight. When this function is used with <code>process_sightings()</code> , this setting from the <code>cruz</code> object will be provided. Note that, although only the

best estimates may be calibrated if specified above (never the highs and lows), the *same* kind of averaging function *is* applied to the highs and lows as is applied to the bests. That is, when `geometric_mean` is `TRUE`, the geometric mean of the highs and the lows is returned. If the best estimates are calibrated, the geometric *weighted* mean will be applied to the highs and lows, using the variance of the calibrated best estimates as weights. If the best estimates are *not* calibrated, the *unweighted* geometric mean is used to estimate the highs, lows, and bests.

`verbose` Boolean, with default `FALSE`, indicating whether or not updates should be printed to the Console.

Value

A modified `cruz` object, in which a `sightings` dataframe (one row for each species in each sighting, with school size estimates averaged and calibrated according to cohort-specific settings) has been added to each `<cohort>` list. The `sightings` dataframe has a new column, `included`, indicating whether or not a sighting will be included in an analysis according to the inclusion criteria specified in cohort-specific settings, as well as a new column, `ss_valid`, indicating whether or not the school size estimate for this sighting is valid and appropriate for use in abundance estimation and/or detection function fitting with a school-size covariate. All columns are described in detail within the documentation for the `LTabundR` function, `process_surveys()`.

<code>process_strata</code>	<i>Process geo-strata</i>
-----------------------------	---------------------------

Description

Determine in/out status of DAS events for each geo-strata provided by settings. This function loops through each stratum dataframe you have provided it in `settings$strata`, formats the stratum, and asks whether each DAS row occurs within it.

This is an internal function typically not called by a user directly. It is the second subroutine called within `process_surveys()`, after `load_das()` and before `format_das()`.

Usage

`process_strata(das, settings, verbose = TRUE)`

Arguments

`das` A `data.frame` of a DAS survey data file, created by `load_das()`.

`settings` A settings object, created by `load_settings()`.

`verbose` Boolean, with default `TRUE`, indicating whether or not updates should be printed to the Console.

Value

A list with two named slots: `settings` and `das`. The `das` slot has the dataframe of your survey data. For each stratum, a column named `stratum_<StratumName>` is added to the `das` dataframe; each row in this column is `TRUE` (included) or `FALSE`.

process_subgroups	<i>Analyze subgroup detections for false killer whales</i>
-------------------	--

Description

Subgroups are found based on DAS event code "G" and associated events. Subgroup sizes are estimated for each phase (based upon the false killer whale protocol), and both arithmetic and geometric means are provided.

This is an internal function typically not called by a user directly. It is the final processing subroutine called within `process_surveys()`, after the subroutine `process_sightings()`.

Usage

```
process_subgroups(cruz, edits = NULL, verbose = TRUE)
```

Arguments

cruz	A cruz object passed from <code>process_surveys()</code> .
edits	An optional input: a list indicating manual adjustments to the subgroup events data. This list can contain elements produced by the LTabundR functions <code>subgroup_populations()</code> or <code>subgroup_edits()</code>, or simply list(s) that the user prepares themselves. Each list element needs to be either a data.frame or a list with the following names: <code>edit</code>, <code>cohort</code>, <code>crusie</code>, and <code>sgid</code>. The remaining names depend on the type of edit (see the documentation for the <code>subgroup_edits()</code> function for details). Edits will be applied in the order in which they are supplied. If this input is supplied, the <code>process_subgroups()</code> function will first process the data using default operations, then redact the events data according to this input. The recommended workflow here is: (1) process your survey data without supplying edits; (2) review the events data using the <code>subgroup_explorer()</code> function; (3) if revisions are needed, stage those edits using <code>subgroup_populations()</code> function, <code>subgroup_edits()</code> function, or manually coded lists. (4) collate those staged edits into a list. (5) re-run <code>process_subgroups()</code>, this time supplying the <code>edits</code> input. See the vignette for details.
verbose	Boolean, with default <code>TRUE</code> , indicating whether or not updates should be printed to the Console.

Value

A finalized cruz object. If subgroups are found, a subgroups slot is added to the analysis list for a cohort. This subgroups slot holds a list with three dataframes:

1. events, in which each row is a school size estimate for a single subgroup during a single phase – 1 or 2 – within a single sighting;
2. subgroups, in which each row is a single phase for a single subgroup, with all school size estimates averaged together (both arithmetically and geometrically);
3. sightings, in which each row is a school size estimate for a single phase for a single sighting, with all subgroup school sizes summed together.

Phase is determined simply according to the OnEffort column. If it is TRUE, Phase is 1; if not, Phase is 2.

process_surveys

Process & prepare Wincruz survey data for analysis.

Description

This function is the main command you will use to begin working with your survey data in R. It takes Wincruz survey data – .DAS file(s) – and user-specified settings to prepare the data for density estimation and/or habitat modeling analyses. This function was designed to be similar to ABUND9, the Fortran program written by Jay Barlow (NOAA SWFSC) for the same purpose.

Usage

```
process_surveys(
  das_file,
  settings,
  edits = NULL,
  process_sightings = TRUE,
  process_subgroups = TRUE,
  save_local = FALSE
)
```

Arguments

das_file	One or more filenames/filepaths to DAS files with survey data, supplied as a character vector. If multiple files are supplied, each file will be processed separately then combined using LTabundR::cruz_combine(). URL's for online DAS repositories are also accepted.
settings	An object representing the output of load_settings(). See that function's documentation for details.

<code>edits</code>	An optional list of staged edits for modifying the DAS data (after reading in; not the actual data files themselves) before proceeding with data processing. These edits must take the form of the input for the LTabundR function <code>das_editor()</code> (see its documentation for details). If edits are supplied, a temporary version of the DAS data will be created (this temporary version will then be deleted at the end of this function's procedure.)
<code>process_sightings</code>	A Boolean, with default TRUE, indicating whether or not sightings should be processed in addition to effort. When troubleshooting effort segments, it could be useful to set this to FALSE to expedite processing time.
<code>process_subgroups</code>	A Boolean, with default TRUE, indicating whether or not subgroups should be found in the survey data and analyzed.
<code>save_local</code>	If TRUE (not the default), the resulting cruz object will be saved in the current working directory as a .RData file. This may be advantageous, so that this function only needs to be run once in order to have the data formatted and ready for analysis. The .RData filename will be the same as <code>das_file</code> (or the first one provided, if multiple are input), except with a different extension.

Details

This function is a wrapper for several subroutines, which are executed in the following order:

1. Read and format the survey data contained in your DAS file using functions developed in the package `swfscDAS` (this step is carried out using the internal LTabundR function `load_das()`).
2. Interpolate the DAS data, if instructed by settings. See `load_survey_settings()` for details.
3. Determine which DAS events occur within the geo-strata provided by the user (using the internal LTabundR function `process_strata()`).
4. Remove invalid entries, determine the ship used in each cruise, calculate the distance transited between each DAS row, and initiate the data structure for the eventual cruz object output. (using the internal LTabundR function `format_das()`).
5. Parse survey effort into "segments", which are used in variance estimation, and determine which segments should be included in the analysis based upon user-specified settings (using the internal LTabundR function `segmentize()`).
6. Process sightings (optional) by determining which should be included in the analysis based upon user-specified settings, and refine school size estimates by calibrating observer estimates and averaging estimates from multiple observers. (using the internal LTabundR function `process_sightings()`).
7. Process subgroup size estimates (optional) for false killer whales (using the internal LTabundR function `process_subgroups()`).
8. The above process is repeated for each DAS file; if there are multiple DAS files, the results are combined using `LTAbundR::cruz_combine()`.
9. Optionally save the result as an RData object to easily pass the processed data to other R scripts.

Value

A cruz object, which is a nested list with the following primary slots:

1. settings, containing the settings list you provided as an argument.
2. strata, containing a dataframe summarizing the geostrata provided (their name and area, in square km).
3. cohorts, containing a named list for each cohort you specified within the settings argument.

Each cohort slot has a similar structure: `cruz$cohorts$<cohort>$<details>`

- The name of each <cohort> slot is drawn from the id slot within that cohort's settings (e.g., `settings$cohorts[[1]]$id`).
- All cohorts have the same three slots for <details>:
 1. segments, a data.frame with metadata for each effort segment (see below).
 2. das, a data.frame of the survey data (see below).
 3. sightings, a data.frame with details for each sighting (see below).
- Some cohorts may also have a fourth <details> slot, subgroups, if subgroups were found in the data for the species specified in the cohort. This slot will contain a list (see below).

segments data structure

A data.frame with metadata for each segment; each row is a segment.

1. Cruise: Cruise number
2. ship: Ship name initials
3. stratum: Stratum designation for this segment
4. seg_id: Unique segment identifier
5. yday: Numeric day of year
6. dist: Distance surveyed in this segment
7. lat1: Start latitude of segment, decimal degrees
8. lon1: Start longitude of segment, decimal degrees
9. DateTime1: Date and time for start of segment, formatted as a `lubridate::datetime` object
10. timestamp1: Numeric timestamp for start of segment (seconds since 00:00:00 UTC on 1 January 1970)
11. lat2: Ending latitude
12. lon2: Ending longitude
13. DateTime2: Date and time of end of segment
14. timestamp2: Numeric timestamp for end of segment
15. mlat: Latitude of middle of segment (i.e., the coordinate for the row of DAS data that is at `nrow(<segment data>) / 2`)
16. mlon: Longitude of middle of segment
17. mDateTime: Date and time of middle of segment
18. mtimestamp: Numeric timestamp of middle of segment

19. use: A decision as to whether or not this segment will be included in the analysis, based upon user-specified criteria in settings.
20. Mode: Effort mode (P for passing or C for closing)
21. EffType: Effort type (S for systematic, N for non-systematic – i.e., off design-based trackline – and F for fine-scale)
22. OnEffort: If TRUE, standard search protocols are in practice
23. ESWsides: Number of sides for which the effective strip width (ESW) will apply. When traveling nearshore, this may be only 1
24. year: Year
25. month: Numeric month
26. day: Numeric day of month
27. min_line: The line number of DAS data at the start of this segment
28. max_line: The final line number
29. n_rows: Number of rows of DAS data in this segment
30. avgBft: Weighted average Beaufort sea state during this segment
31. avgSwellHght: Weighted average Swell Height, in feet, during this segment
32. avgHorizSun: Weighted average horizontal sun angle, corresponding to a clock face, during this segment
33. avgVertSun: Weighted average vertical sun angle (12 = overhead, 1-3 = at the horizon) during this segment
34. avgGlare: Weighted average Glare status during this segment
35. avgVis: Weighted average visibility (defines as the distance, in nautical miles, at which a dolphin could be seen surfacing with the water, not the sky, as the background) during this segment
36. avgCourse: Weighted average ship heading during this segment
37. avgSpdKt: Weighted average speed, in knots, during this segment

das data structure

The data.frame of DAS survey data, as read and formatted by `swfscDAS::das_read()` and `swfscDAS::das_process()`. See the latter function documentation for details on columns. We have added the following columns during the preparation of the `cruz` object:

- `stratum_<stratum name>` A Boolean indicating whether or not this row of data occurs within this geostratum polygon. There will be a column like this for each geostratum provided in your settings object.
- `year` Year
- `month` Numeric month
- `day` Numeric day of month
- `yday` Numeric day of year
- `km_int` Distance, in kilometers, between this row of data and the next. See documentation for `process_km()` for details.
- `km_cum` Cumulative distance traveled up to this point in the DAS data.

- ship Ship name initials
- stratum Final stratum designation, decided based upon user-specific settings.
- seg_id Identifier for the segment containing this row of data.
- use: A Boolean decision as to whether or not this segment will be included in the analysis, based upon user-specified criteria in settings.

sightings data structure

The `data.frame` of processed sightings, as prepared by `swfscDAS::das_sight(return.format = 'complete')`. See that function's documentation for details on columns. Note that the unique sighting identifier can be found in column `SightNodaily`, not `SightNo`. That function returns up to several rows for each sighting (DAS event codes S, s, A, 1, 2, ..., etc.). We have processed the result further such that each row represents the school size estimate for a single species within a single sighting. For example, single-species sightings will always have just one row. We have also added the following columns during the preparation of the `cruz` object:

- See the columns added to the `das` slot above; those have all been propagated to this `sightings` table.
- species: species code (character string) for the species represented by this row.
- best: Best estimate of school size (calibrated and averaged across observer estimates according to settings)
- low: Low estimate of school size (ditto)
- high: High estimate of school size (ditto)
- prob: If TRUE, this species code is a probable identification (DAS event code ?).
- mixed: If TRUE, this species occurred in a mixed-species school.
- ss_tot: Total school size (calibrated and averaged across observer estimates according to settings)
- ss_percent: Percent of the school comprised by this species (averaged across observer estimates).
- n_sp: Number of species in this sighting.
- n_obs: Number of observers who contributed a school size estimate for this species.
- n_best: Number of valid best estimates of school size.
- n_low: Number of valid low estimates.
- n_high: Number of valid high estimates.
- calibr: Boolean indicating whether school size calibration was applied.
- ss_valid: Boolean indicating whether or not the school size estimate for this sighting is valid and appropriate for use in abundance estimation and/or detection function fitting with a school-size covariate.
- included: Boolean indicating whether the sightings should be included in the analysis based on the specified settings. Any sighting with `use == FALSE` will also have `included == FALSE`, but it *is* possible for sightings to have `use == TRUE` with `included == FALSE`. For example, if the setting `abeam_sightings` is set to FALSE, a sighting with a bearing angle beyond the ship's beam can be excluded from the analysis (`included == FALSE`) even though the effort segment it occurs within will still be used (`use == TRUE`).

subgroups data structure

If subgroup events (DAS event code G) are found pertaining to the species in your cohort, this slot will have a list with three slots:

1. **sightings:** A `data.frame` in which each row is a school size estimate for a single phase for a single sighting, with all subgroup school sizes summed together. Columns:
 - (a) `Cruise`
 - (b) `Date`
 - (c) `SightNo`
 - (d) `Phase`
 - (e) `DateTime` Mean date and time of estimates of this sighting
 - (f) `Lat` (same – mean)
 - (g) `Lon` (same – mean)
 - (h) `Species`
 - (i) `Angle` (same – mean)
 - (j) `RadDist` (same – mean)
 - (k) `seg_id` Identifier for the segment containing this row of data.
 - (l) `PerpDist` (same – mean)
 - (m) `GSBest` Sum of arithmetic means of best estimates of subgroups
 - (n) `GSBest_geom` Sum of geometric means
 - (o) `EffType` Type of effort (systematic, nonsystematic, or fine-scale)
 - (p) `OnEffort` Whether or not standard search protocols are in use (TRUE or FALSE).
 - (q) `use` Whether the segment on which this sighting occurred meets analysis inclusion criteria.
 - (r) `stratum_[stratum name]` A set of columns, one for each geostratum in the `cruz` object settings file, indicating whether or not this location occurs within each geostratum.
 - (s) `stratum` The geostratum to which this location was assigned, based upon cohort settings.
2. **subgroups:** A `data.frame` in which each row is a single phase for a single subgroup, with all school size estimates averaged together (both arithmetically and geometrically). Columns:
 - (a) `Cruise`
 - (b) `Date`
 - (c) `DateTime` First date and time of estimates of this subgroup
 - (d) `Lat` (same – first data point for estimates of this subgroup)
 - (e) `Lon` (same)
 - (f) `OnEffort` (same)
 - (g) `EffType` (same)
 - (h) `SightNo` (same)
 - (i) `Species` (same)
 - (j) `SubGrp` (same)
 - (k) `Angle` Mean angle to this subgroup
 - (l) `RadDist` Mean radial distance to this subgroup
 - (m) `seg_id` Identifier for the segment containing this row of data.
 - (n) `PerpDist` Mean perpendicular distance to this subgroup

- (o) GSBest Arithmetic mean of best estimates of this subgroup. If no best estimates are given, the GSL value will be used here.
 - (p) GSH Arithmetic mean of high estimates of this subgroup
 - (q) GSL Arithmetic mean of low estimates of this subgroup
 - (r) GSBest_geom Geometric mean of best
 - (s) GSH_geom Geometric mean of high
 - (t) GSL_geom Geometric mean of low
 - (u) Phase Phase (1 or 2)
 - (v) use Whether the segment on which this sighting occurred meets analysis inclusion criteria.
 - (w) stratum_[stratum name] A set of columns, one for each geostratum in the cruz object settings file, indicating whether or not this location occurs within each geostratum.
 - (x) stratum The geostratum to which this location was assigned, based upon cohort settings.
3. events: A data.frame in which each row is single subgroup size estimate from a single observer for a single phase (Phase 1 – on effort / passing mode – or Phase 2 – off effort / closing mode). This is effectively the "raw" subgroup data. Columns:
- (a) Cruise Cruise number
 - (b) Date Date, in format YYYY-MM-DD
 - (c) DateTime Date and time, in format YYYY-MM-DD HH:MM:SS
 - (d) Lat Latitude, in decimal degrees
 - (e) Lon Longitude, in decimal degrees
 - (f) OnEffort If TRUE, standard search protocols are in use. If FALSE, non-standard protocols (or no searching at all).
 - (g) EffType Effort type (S for systematic, N for non-systematic, and F for fine-scale)
 - (h) SightNo Sighting number of the day (the count resets every day)
 - (i) Species Species code (character string)
 - (j) Line Line number in DAS data
 - (k) SubGrp Subgroup identifier code
 - (l) Event Count of estimates for this subgroup-phase
 - (m) Obs Observer code
 - (n) GSBest Best estimate of subgroup size
 - (o) GSH High estimate of subgroup size
 - (p) GSL Low estimate of subgroup size
 - (q) Angle Angle between bow and subgroup
 - (r) RadDist Radial distance to subgroup, in km
 - (s) use Whether the segment on which this sighting occurred meets analysis inclusion criteria.
 - (t) stratum_[stratum name] A set of columns, one for each geostratum in the cruz object settings file, indicating whether or not this location occurs within each geostratum.
 - (u) stratum The geostratum to which this location was assigned, based upon cohort settings.
 - (v) PerpDist Perpendicular distance to subgroup from the trackline represented by the ship heading, in km
 - (w) sgid Unique subgroup identifier

- (x) `sitid` Unique sighting identifier
- (y) `phase` Phase in protocol; all `OnEffort == TRUE` estimates are Phase 1; all `OnEffort == FALSE` estimates are Phase 2.

This `cruz` object can be carried forward into data exploration (e.g., see the `LTabundR` function `cruz_explorer()`), analyses (e.g., see the `LTabundR` function `lta()`), or passed to mapping functions (see the `LTabundR` functions that begin with `map_...`) or summary functions (see the `LTabundR` functions that begin with `summarize_...`).

<code>read_polygon</code>	<i>Read in a DAT file with polygon(s)</i>
---------------------------	---

Description

This is an internal function typically not called by a user directly. This function reads in a .DAT file with coordinates for one or more polygons, returning a list.

Usage

```
read_polygon(dat_file)
```

Arguments

`dat_file` Filepath to a .DAT file containing coordinates for one or more polygons.

Value

A named list:

1. `polygons` A dataframe summarizing the polygons contained in the .DAT file. It has four columns: `name` (polygon name), `area` (polygon area, if provided), `name_i` (the index of the polygon in the file), and `area_i` (the index of the area provided in the file).
2. `coordinates` A list with a slot for each polygon contained within the .DAT file. Each polygon is a dataframe with two columns: `Lon` and `Lat`, with coordinates formatted as provided by the .DAT file.
3. `dat` The raw DAT file, as it was read by R, returned as a dataframe.

region_das	<i>Find DAS rows occurring within a region / regions</i>
------------	--

Description

This is an internal function, typically not called by the user.

Usage

```
region_das(das, regions)
```

Arguments

- das The DAS table from a cruz object.
- regions A character vector of geostrata whose segments you want.

Value

A vector of line_num values whose data occur within any of regions, which can be used to filter segments and sightings.

region_segments	<i>Find segments occurring within a region / regions</i>
-----------------	--

Description

This is an internal function, typically not called by the user.

Usage

```
region_segments(das, regions)
```

Arguments

- das The DAS table from a cruz object.
- regions A character vector of geostrata whose segments you want.

Value

A vector of segment IDs whose data occur within regions, which can be used to filter segments and sightings by the seg_id column.

segmentize

*Segmentize survey effort***Description**

This function chops DAS effort into discrete sections for the purposes of estimating the variance of the abundance estimate.

This is an internal function typically not called by a user directly. It is the fourth subroutine called within `process_surveys()`, after `format_das()` and before `process_sightings()`. However, arguments are provided that allow you to call this function directly and override settings so that you can explore how segmentizing works.

Usage

```
segmentize(
  cruz,
  segment_method = NULL,
  segment_target_km = NULL,
  segment_max_interval = NULL,
  segment_remainder_handling = NULL,
  beaufort_range = NULL,
  distance_types = NULL,
  distance_modes = NULL,
  distance_on_off = NULL,
  seed = NULL,
  to_plot = TRUE,
  debug_mode = FALSE,
  verbose = FALSE
)
```

Arguments

- | | |
|-------------------|--|
| cruz | A nascent cruz object, which is a list produced by <code>format_das()</code> , or a fully processed cruz object if you want to experiment with segmentizing alternatives. |
| segment_method | This and the remainder of inputs allow you to override the settings contained within the cruz object. Leaving them as NULL means the original settings will be used. For <code>segment_method</code> , the two method options are "day" – all effort within the same Cruise-Stratum-Year-Effort scenario (i.e., an effort bloc) will be binned into segments by calendar date – and "equallength" – effort within each unique effort bloc will be divided into segments of approximately equal length. |
| segment_target_km | If segmentizing by "equallength", this field allows you to specify what that target length is, in km. |

segment_max_interval

If segmentizing by "equallength", this setting allows you to specify the time gaps in effort (in hours) that are allowed to be contained within a single segment. For example, if your goal is a few large segments of equal length (e.g., 150-km segments, for bootstrap estimation of density variance), you are probably willing for discrete periods of effort to be concatenated into a single segment, even if the gaps between effort are as large as 1 or 2 days, in which case you would set `segment_max_interval` to 24 or 48 (hours), respectively. However, if your goal is many smaller segments (e.g., 5-km segments, for habitat modeling), you want to ensure that effort is contiguous so that segment locations can be accurately related to environmental variables, in which case you would set `segment_max_interval` to be very small (e.g., .2 hours, or 12 minutes). Setting this interval to a small number, such as 0.2, also allows the segmentizing function to overlook momentary breaks in effort, such as when an unofficial observer logs a sighting.

segment_remainder_handling

If segmentizing by "equallength", periods of effectively-contiguous effort (as specified by `segment_max_interval`) are unlikely to be perfectly divisible by your `segment_target_km`; there is going to be a remainder. You can handle this remainder in three ways: (1) "disperse" allows the function to adjust `segment_target_km` so that there is in fact no remainder, effectively dispersing the remainder evenly across all segments within that period of contiguous effort; (2) "append" asks the function to append the remainder to a randomly selected segment, such that most segments are the target length with the exception of one longer one; or (3) "segment" asks the function to simply place the remainder in its own segment, placed randomly within the period of contiguous effort. This setting also has a second layer of versatility, because it can accept a one- or two-element character vector. If a two-element vector is provided (e.g., `c("append", "segment")`), the first element will be used in the event that the remainder is less than or equal to half your `segment_target_km`; if the remainder is more than half that target length, the second element will be used. This feature allows for replication of the segmentizing methods in Becker et al. (2010).

beaufort_range A numeric vector indicating the Beaufort sea states (0 - 7) to be accepted within usable segments.

distance_types A character vector of the effort types that will be included in detection function estimation, and therefore considered in effort segmentizing. Accepted values are "S" (systematic/standard effort), "F" (fine-scale effort), and "N" (non-systematic/non-standard effort, in which systematic protocols are being used but effort is not occurring along design-based transect routes).

distance_modes The effort modes that will be included in detection function estimation, and therefore considered in effort segmentizing. Accepted values are "P" (passing) and "C" (closing)

distance_on_off

The value(s) of `OnEffort` (On Effort is TRUE, Off Effort is FALSE) that will be included in detection function estimation, and therefore considered in effort segmentizing.

seed Supply a number to ensure that segment chopping is exactly reproducible. Some of the remainder handling methods (namely segment and append) will place the

	remainder to a randomly selected segment. Supplying a number here will ensure the remainder goes in the same place with each run. If left NULL, the segment breaks are liable to differ each time this function is run.
to_plot	Boolean, with default FALSE, indicating whether or not histograms showing segment lengths should be produced.
debug_mode	Boolean, with default FALSE, indicating whether details should be printed to the Console that facilitate debugging.
verbose	Boolean, with default FALSE, indicating whether or not updates should be printed to the Console.

Value

A modified cruz object, in which each cohort slot contains a list with the following slots: das (with two new columns, seg_id, indicating the segment corresponding to each DAS row; and use, indicating whether or not (TRUE or FALSE) the segment is to be included in the analysis.) and segments (a dataframe summarizing each segment).

ships	<i>Ship codes</i>
-------	-------------------

Description

Ship codes

Usage

ships

Format

An object of class data.frame with 66 rows and 3 columns.

Details

A data.frame matching the ship name initials to the NOAA-NFMS cruise number.

species_codes	<i>Species codes used in WinCRUZ</i>
---------------	--------------------------------------

Description

Species codes used in WinCRUZ

Usage

```
species_codes
```

Format

A data.frame with 164 rows and 5 variables:

code Code used in WinCRUZ

short_name Abbreviated name, in all CAPS

scientific_name Latin Genus species

common Most common name

description Alternative common names

Details

A data.frame containing species codes and associated identifiers.

species_translator	<i>Species name/code search engine</i>
--------------------	--

Description

Retrieve common and scientific names for a species based on their code number or short name, or vice versa: retrieve the species code based on common or scientific name.

Usage

```
species_translator(
  id,
  codes = NULL,
  match_by_code = TRUE,
  match_by_short_name = TRUE,
  match_by_latin = TRUE,
  match_by_common = TRUE
)
```

Arguments

id	Your search query term; can be a character or numeric. The function will return partial matches and is case insensitive.
codes	A data.frame of species codes, with the same column structure as the LTabundR dataset data(species_codes). If not provided by the user, that built-in dataset will be provided.
match_by_code	The remaining inputs are a way to toggle on/off certain methods for searching for matches. This input will try to match your search query to species codes, e.g., "033" for the false killer whale.
match_by_short_name	Try matching search query to "short name" species codes, e.g., "MESOP_PERU" for the pygmy beaked whale.
match_by_latin	Try matching search query to scientific name.
match_by_common	Try matching search query to common name.

Value

A data.frame with all viable patches to your search query.

Examples

```
species_translator('046') # match by species code
species_translator('46') # partial matches work
species_translator(46) # numeric codes work
species_translator('RM_WHA') # partial short names work
species_translator('Physeter') # latin names work
species_translator('orca') # common names work + partial matches + case insensitive
species_translator('orca', match_by_latin=FALSE) # toggle match modes
```

strata_area

Combine geostrata and compute area

Description

This function allows for complex combinations / subtractions of geostratum polygons and calculates the resulting area before and after the removal of dry land. Diagnostic plots are provided that ensure the result is correct.

Usage

```
strata_area(
  strata_all,
  strata_keep,
  strata_remove = NULL,
```

```

remove_land = TRUE,
toplot = TRUE,
verbose = TRUE
)

```

Arguments

<code>strata_all</code>	A named list in which each slot is a <code>data.frame</code> of coordinates for a geostratum polygon. Each <code>data.frame</code> must have <code>Lat</code> and <code>Lon</code> as the first two columns, providing coordinates in decimal degrees in which South and West coordinates are negative. Other columns are allowed, but the first two need to be <code>Lon</code> and <code>Lat</code> . The name of the slot holding the <code>data.frame</code> will be used as a reference name for the stratum. For an example of formatting, see <code>data(strata_cnp)</code> . If you are working with a standard NOAA survey region, such as the Central North Pacific (CNP), Eastern Tropical Pacific (ETP), or California Current System (CCS), you can use built-in polygons available in <code>data(strata_cnp)</code> , <code>data(strata_etp)</code> , or <code>data(strata_ccs)</code> , respectively. To explore and/or select strata contained within those built-in datasets, use the functions <code>strata_explore()</code> and <code>strata_select()</code> .
<code>strata_keep</code>	A character vector of the names of the geostrata within <code>strata_all</code> that you want to use
<code>strata_remove</code>	A character vector of the names of the geostrata you want to remove. For example, perhaps you want to keep the <code>HI_EEZ</code> stratum but remove insular stock boundaries.
<code>remove_land</code>	A Boolean, with default <code>TRUE</code> , indicating whether or not you want to remove land from the resulting survey area.
<code>toplot</code>	If <code>TRUE</code> (the default), diagnostic maps will be plotted for your review.
<code>verbose</code>	If <code>TRUE</code> (the default), status updates will be printed to the Console.

Value

A list with the following slots:

1. `km2` The area of the resulting survey area polygon, in square km.
2. `km2_keep` The area of the `strata_keep` polygons, after combining them to account for any overlapping portions of the polygons, but before removing land.
3. `km2_remove` The area of the `strata_remove` polygons, if any were provided, after combining them to account for any overlapping portions of the polygons, but before removing land.
4. `km2_with_land` The area of the resulting survey area polygon *before* removing land.
5. `sf` The `sf` object of the resulting polygon.

Note that if `toplot` is `TRUE` (the default), a diagnostic plot will also be produced.

strata_ccs	<i>Strata CCS</i>
------------	-------------------

Description

Strata CCS

Usage

strata_ccs

Format

An object of class list of length 5.

Details

Typical geostrata used in the California Current System region.

strata_cnp	<i>Strata CNP</i>
------------	-------------------

Description

Strata CNP

Usage

strata_cnp

Format

An object of class list of length 12.

Details

Typical geostrata used in the Central North Pacific region.

strata_etp	<i>Strata ETP</i>
------------	-------------------

Description

Strata ETP

Usage

strata_etp

Format

An object of class list of length 70.

Details

Typical geostrata used in the Eastern Tropical Pacific region.

strata_explore	<i>Explore the package's strata polygon gallery</i>
----------------	---

Description

The LTabundR package comes with several built-in datasets of geographic strata that are commonly used in NOAA/NMFS surveys. The functions strata_explore() and strata_select() were developed to help you explore those built-in options.

strata_explore() launches a leaflet map displaying the built-in geostrata datasets available for your use in the common NOAA/NMFS survey regions. Each stratum displayed as a separate layer that can be hidden or hovered over to get details.

Usage

```
strata_explore(  
  region,  
  start_at = 1,  
  strata_color = "darkblue",  
  begin_hidden = FALSE  
)
```

Arguments

region	Name of the survey region of interest. Options currently supported: "cnp" (the default; Central North Pacific), "ccs" (California Current System), and "etp" (Eastern Tropical Pacific).#
start_at	A number indicating the geostratum index you wish to begin at, if you know you aren't interested in those early on in the list. For example, the ETP has over 70 built-in geostrata available!
strata_color	Color of geostratum.
begin_hidden	If TRUE (not the default), the strata will begin hidden, i.e., unchecked, so that you have to opt to visualize each stratum individually. This can be handy for regions with many geostrata to choose from, such as the ETP, which has 70 geostrata!

Value

An interactive leaflet map. The name of the geostratum, and the numeric identifier for it, are printed in a legend as well as in a pop-up box that appears when you hover over a stratum. Use this identifier as an argument in the companion function, `strata_select()`, to access the datasets of coordinates underlying the polygons displayed.

strata_segments	<i>Display/map stratum assignments for segments</i>
-----------------	---

Description

A simple diagnostic function, usually not used by most analysts.

Usage

```
strata_segments(cruz, cohort = 1, plot_title = "Stratum assignments")
```

Arguments

cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).
cohort	The cohort whose data you would like to display, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.
plot_title	Optional title for plot.

Value

A `ggplot2` object.

strata_select	<i>Select strata polygons for standard survey regions</i>
---------------	---

Description

The LTabundR package comes with several built-in datasets of geographic strata that are commonly used in NOAA/NMFS surveys. The functions `strata_explore()` and `strata_select()` were developed to help you explore those built-in options.

Those built-in datasets come in the form of a list, in which each slot is a separate geostratum. `strata_select()` allows you to "grab" the underlying data (i.e., the `data.frame` of Lat/Long coordinates) for one of the geostrata displayed by `strata_explore()`.

Usage

```
strata_select(selections, region)
```

Arguments

<code>selections</code>	Numeric index of the geostratum you wish to "grab" from a list of geostrata. This index is printed in the legend produced by <code>strata_explore()</code> .
<code>region</code>	Name of the survey region pertaining to your selection (the same region you passed to <code>strata_explore()</code>). Options currently supported: "cnp" (the default; Central North Pacific), "ccs" (California Current System), and "etp" (Eastern Tropical Pacific).

Value

A `data.frame` of coordinates for the boundary of a geostratum. You can pass this object to the `strata` argument for `load_settings()`.

subgroup_edits	<i>Stage edits to subgroups (e.g., phase and population assignments)</i>
----------------	--

Description

Stage edits to subgroups (e.g., phase and population assignments)

Usage

```
subgroup_edits(
  cohort,
  sgid,
  exclude = NULL,
  ObsStd = NULL,
```

```

    phase = NULL,
    population = NULL,
    pop_prob = NULL
  )

```

Arguments

cohort	The cohort whose data will be edited, provided as the number or name of the slot in <code>cruz\$cohorts</code> to be referenced.
sgid	A character vector of subgroup IDs whose data will be edited.
exclude	A Boolean vector, with length of either 1 or the same as <code>sgid</code> , indicating which <code>sgid</code> 's to erase: TRUE means exclude, FALSE means keep.
ObsStd	A Boolean vector, with length of either 1 or the same as <code>sgid</code> , indicating whether or not the observer of this subgroup was a standard/primary observer.
phase	A numeric vector, with length of either 1 or the same as <code>sgid</code> , indicating the new phase to assign the <code>sgid</code> 's. If you wish to remove a phase assignment for a <code>sgid</code> , use NA.
population	A character vector, with length of either 1 or the same as <code>sgid</code> , indicating the new population to assign the <code>sgid</code> 's. If you want a <code>sgid</code> to be eligible for multiple populations, separate the population names with a semicolon. Both this input and <code>pop_prob</code> needs to not be NULL in order for this input to be recognized.
pop_prob	A numeric vector, with length of either 1 or the same as <code>sgid</code> and values ranging between 0 and 1, indicating the probability of each population assignment for the <code>sgid</code> 's. If you want a <code>sgid</code> to be eligible for multiple populations, separate the probabilities by a semicolon. Both this input and <code>population</code> needs to not be NULL in order for this input to be recognized.

Value

A list of staged edits that will be accepted by the `edit`s input in `process_subgroups()`.

subgroup_events	<i>Find subgroup size estimates within a DAS file</i>
-----------------	---

Description

This is an internal function typically not called by a user directly. It is called in `process_subgroups()`, which is an internal function called in `process_surveys()`.

Usage

```
subgroup_events(das, species_filter = "033")
```

Arguments

- `das` A dataframe of DAS data, formatted by the LTabundR function `load_das()`.
- `species_filter` Species code(s), as a character vector, if you only want subgroup data for certain species. The default is for false killer whales, *Pseudorca crassidens*, for whom the subgroup data collection protocol was developed (see [Bradford et al. 2018](#)).

Value

A `data.frame` in which each row is a single school size estimate for a single subgroup within a single phase of a single sighting (effectively, the 'raw' data for subgroups within the `das` data).

<code>subgroup_explorer</code>	<i>Simple explorer for subgroups data</i>
--------------------------------	---

Description

Simple explorer for subgroups data

Usage

```
subgroup_explorer(cruz, cohort = "pseudorca")
```

Arguments

- `cruz` Your `cruz` object (produced from LTabundR: `:process_surveys()`).
- `cohort` The cohort whose data you would like to review, provided as the number or name of the slot in `cruz$cohorts` to be referenced.

Value

A Shiny app to explore subgroup event data for a cohort within a `cruz` object.

<code>subgroup_phases</code>	<i>Review & stage edits to subgroup phase assignments</i>
------------------------------	---

Description

Review & stage edits to subgroup phase assignments

Usage

```
subgroup_phases(cruz, cohort)
```

Arguments

cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).
cohort	The cohort whose subgroup phases you would like to review and stage edits for, provided as either the cohort name (e.g., "all" or "pseudorca") or a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.

Value

A Shiny app that allows you to make manually stage revisions to the phase assigned to each subgroup (this is in reference to the protocol developed for false killer whales by NOAA PIFSC). Those phases were assigned automatically during the `process_surveys()` routine; this is a way for you to review those assignments and prepare a set of revisions. When you close the app, those revisions are returned as a `list()`, which you can then save to file and/or pass to `LTabundR::subgroup_edit()`, which will implement the revisions by modifying the data in your cruz object. This function, `subgroup_phases()`, does not edit the cruz object in any form.

subgroup_populations *Assign subgroups to populations based on polygons*

Description

Assign subgroups to populations based on polygons

Usage

```
subgroup_populations(
  populations,
  cruz,
  cohort = "pseudorca",
  default_pop = "pelagic",
  verbose = TRUE
)
```

Arguments

populations	A named list in which each slot is a <code>data.frame</code> of coordinates for a geostratum polygon. Each <code>data.frame</code> must have <code>Lon</code> and <code>Lat</code> as the first two columns, providing coordinates in decimal degrees in which West and South coordinates are negative. It is acceptable if vertices in the eastern hemisphere are described using negative longitudes below -180, e.g., -185. Other columns are allowed, but the first two need to be <code>Lon</code> and <code>Lat</code> . The name of the slot holding the <code>data.frame</code> will be used as a reference name for the stratum. Note that if coordinates in your data or in your collection of strata span the International Date Line (IDL) such that some longitudes are positive and some are negative, during data processing all longitudes will be coerced to negative degrees West.
cruz	Your cruz object (produced from <code>LTabundR::process_surveys()</code>).

cohort	The cohort whose data you would like to use, provided as the number or name of the slot in <code>cruz\$cohorts</code> to be referenced.
default_pop	desc
verbose	Boolean; print updates to console?

Value

A data.frame of population assignments for each subgroup event, formatted to be accepted as edits as an input in `process_subgroups()`.

subgroup_sightings	<i>Summarize sightings (for subgroup species) based on subgroups</i>
--------------------	--

Description

This is an internal function, not typically called by the user. It is called during the `process_subgroups()` routine within `process_surveys()`, and may also be called if the user manually edits subgroup phase assignments using the `subgroup_phases()` function. It takes the result of the function `subgroup_subgroups()`.

Usage

```
subgroup_sightings(subgroups)
```

Arguments

subgroups	The output of <code>subgroup_subgroups()</code> .
-----------	---

Value

A data.frame with one row for each phase of each sighting.

subgroup_subgroups	<i>Summarize subgroups based on subgroup events</i>
--------------------	---

Description

This is an internal function, not typically called by the user. It is called during the `process_subgroups()` routine within `process_surveys()`, and may also be called if the user manually edits subgroup phase assignments using the `subgroup_phases()` function. It takes the result of the function `subgroup_events()`.

Usage

```
subgroup_subgroups(events)
```

Arguments

events The output of subgroup_events().

Value

A data.frame with one row for each subgroup in each phase of a sighting.

summarize_bft	<i>Summarize Wincruz by Beaufort Sea State</i>
---------------	--

Description

Calculate the distance and proportion of effort within each Beaufort Sea State within your Wincruz survey data.

Usage

```
summarize_bft(cruz, use_only = TRUE, cohort = 1)
```

Arguments

cruz Your cruz object (produced from LTabundR::process_surveys()).

use_only If TRUE (the default), sea states will only be summarized for effort where use=TRUE, which is the effort that will be used in detection function model fitting. If FALSE, all effort will be summarized.

cohort The cohort whose data you would like to summarize, provided as a number indicating which slot in cruz\$cohorts should be referenced.

Value

A list with various summary tables:

1. overall: Overall summary of effort, parsed by Beaufort sea state, for the entire dataset. Three columns: Bft, km, prop (the fraction of effort for this sea state).
2. by_year: Same table as overall, this time parsed by year.
3. by_cruise: Same table as overall, this time parsed by geostratum.
4. details: Same table as overall, this time parsed by each unique Cruise-year-stratum combination.

summarize_distances	<i>Summarize sighting distances in a table</i>
---------------------	--

Description

Summarize sighting distances in a table

Usage

```
summarize_distances(
  distances,
  distance_range = c(0, 10),
  distance_interval = 0.5
)
```

Arguments

distances Perpendicular distances to sightings, usually in km.

distance_range The range of distances for which to summarize detection distances.

distance_interval The interval of distances to summarize in the table.

Value

A dataframe with a row for each distance interval, summarizing the number of sightings, etc.

summarize_effort	<i>Summarize Wincruz survey effort</i>
------------------	--

Description

Inventory and summarize survey effort within a LTabundR cruz object in various ways.

Usage

```
summarize_effort(cruz, cohort = 1)
```

Arguments

cruz Your cruz object (produced from LTabundR::process_surveys()).

cohort The cohort whose data you would like to summarize, provided as a number indicating which slot in cruz\$cohorts should be referenced.

Value

A list with various summary tables:

1. `total`: Grand total distance and days surveyed. Two columns: km, days.
2. `total_by_year`: Total distance and days surveyed for each year. Three columns: year, km, days.
3. `total_by_cruise`: Total distance and days surveyed for each cruise. Four columns: Cruise, year, km, days.
4. `total_by_effort`: Total distance and days surveyed, grouped by segments that will be included in the analysis and those that won't. Five columns: Cruise, year, use (used for analysis or not), km, and days.
5. `total_by_stratum`: Total distances and days surveyed within each stratum, again grouped by segments that will be included in the analysis and those that won't. Six columns: Cruise, year, stratum, use, km, days.

<code>summarize_sightings</code>	<i>Inventory sightings and species counts from a Wincruz survey</i>
----------------------------------	---

Description

Inventory and summarize sightings within a `LTabundR` `cruz` object in various ways.

Usage

```
summarize_sightings(cruz, cohort = 1)
```

Arguments

<code>cruz</code>	Your <code>cruz</code> object (produced from <code>LTabundR::process_surveys()</code>).
<code>cohort</code>	The cohort whose data you would like to map, provided as a number indicating which slot in <code>cruz\$cohorts</code> should be referenced.

Value

A list with various summary tables. In each table, each row is a count for a single species code.

1. `simple_totals`: includes all sightings, even if they will not be included in analysis. Seven columns: `code` (species code), `short_name`, `scientific_name`, `common_name`, `n` (number of sightings), `ss_mean` (mean school size), `ss_sd` (standard deviation of school size).
2. `analysis_totals`: only includes sightings that meet all inclusion criteria for the analysis. Same columns as `simple_totals`.
3. `stratum_simple_totals`: includes all sightings, even if they will not be included in analysis, parsed by each geostratum. Same columns as `simple_totals`, with the addition of `year`, `Cruise`, and `stratum`.
4. `stratum_analysis_totals`: only includes sightings that meet all inclusion criteria for the analysis, parsed by each geostratum. Same columns as `simple_totals`, with the addition of `year`, `Cruise`, and `stratum`.

summarize_species	<i>Summarize sightings for a species (or several)</i>
-------------------	---

Description

Summarize sightings for a species (or several)

Usage

```
summarize_species(
  spp,
  cruz,
  cohort = 1,
  filter_to_regions = NULL,
  exclude_regions = NULL,
  distance_restrict = TRUE,
  distance_range = c(0, 10),
  distance_interval = 0.5
)
```

Arguments

spp	A character vector with codes for the species you wish to summarize.
cruz	Your cruz object (produced from LTabundR: :process_surveys()).
cohort	The cohort whose data you would like to map, provided as a number indicating which slot in cruz\$cohorts should be referenced.
filter_to_regions	Regions (geostrata) to filter the results down to. Use cruz\$strata to see the options available to you.
exclude_regions	Regions (geostrata) to exclude from the results. This can be helpful if, for example, you wish to summarize sightings for a large region but not for regions nested within it, such as insular geostrata.
distance_restrict	If TRUE, detection distances will only be shown for sightings that will be included in analysis (included == TRUE).
distance_range	The range of distances for which to summarize detection distances.
distance_interval	The interval of distances to summarize.

Value

A list with the following named slots:

1. species: A dataframe with the codes, common names, and scientific name(s) for the specified species (as found in data(species_codes))

2. `n_total`: Total number of sightings (mixed-species are only counted once; this is true for all other slots here as well).
3. `n_analysis`: Number of sightings that will qualify for inclusion in the analysis (`included == TRUE`).
4. `school_size`: Dataframe with summary metrics of school size for each species.
5. `yearly_total`: Dataframe with counts and school sizes metrics for all sightings, parsed by year of sighting.
6. `yearly_analysis`: Dataframe with counts and school size metrics for sightings included in the analysis, parsed by year of sighting.
7. `regional_total`: Dataframe with counts and school sizes metrics for all sightings, parsed by regional geostratum.
8. `regional_analysis`: Dataframe with counts and school size metrics for sightings included in the analysis, parsed by regional geostratum.
9. `detection_distances`: A dataframe with sighting counts for various perpendicular distances (in KM) (systematic sightings only). The `percent_beyond` column can be used to identify appropriate truncation distances for this species group.
10. `sightings`: Dataframe with all sightings information for this species group.

Index

* datasets

- barlow_2015, [6](#)
- cnp_150km_1986_2020, [7](#)
- cnp_1986_2020_edits, [10](#)
- coastline, [11](#)
- eez, [24](#)
- eez_ccs, [24](#)
- eez_hawaii, [25](#)
- example_cruz, [26](#)
- example_settings, [27](#)
- g0_results, [36](#)
- grp_coeff_abund, [45](#)
- grp_coeff_gerrodetette, [45](#)
- grp_ops_abund, [46](#)
- grp_ops_gerrodetette, [46](#)
- land, [50](#)
- lta_result, [75](#)
- noaa_10km_1986_2020, [84](#)
- ships, [103](#)
- species_codes, [104](#)
- strata_ccs, [107](#)
- strata_cnp, [107](#)
- strata_etp, [108](#)

abundance, [4](#)

barlow_2015, [6](#)

bca, [7](#)

cnp_150km_1986_2020, [7](#)

cnp_1986_2020_edits, [10](#)

coastline, [11](#)

coordinates_in_strata, [11](#)

cruz_combine, [12](#)

cruz_explorer, [12](#)

cruz_structure, [13](#)

das_editor, [13](#)

das_editor_tofile, [14](#)

das_format, [15](#)

das_inspector, [16](#)

das_interpolate, [17](#)

das_load, [17](#)

das_readtext, [18](#)

das_time, [19](#)

df_curve, [19](#)

df_fit, [20](#)

df_plot, [22](#)

eez, [24](#)

eez_ccs, [24](#)

eez_hawaii, [25](#)

er_simulator, [25](#)

example_cruz, [26](#)

example_settings, [27](#)

filter_cohort, [28](#)

filter_cruz, [30](#)

g0_combine, [32](#)

g0_model, [32](#)

g0_optimize, [35](#)

g0_plot, [35](#)

g0_results, [36](#)

g0_table, [37](#)

g0_weighted, [38](#)

gg_effort, [40](#)

gg_geostratum, [41](#)

gg_sightings, [42](#)

ggplot_idl, [43](#)

grp_calibrate_abund, [44](#)

grp_coeff_abund, [45](#)

grp_coeff_gerrodetette, [45](#)

grp_ops_abund, [46](#)

grp_ops_gerrodetette, [46](#)

grp_recalibrate, [47](#)

grp_size, [48](#)

idl_coerce, [49](#)

land, [50](#)

load_cohort_settings, 50
load_settings, 54
load_survey_settings, 55
lta, 58
lta(), 79
lta_checks, 66
lta_ci, 67
lta_combine, 68
lta_delta_method, 69
lta_destratify, 69
lta_diagnostics, 70
lta_enlist, 71
lta_estimates, 72
lta_plot, 73
lta_pool, 74
lta_report, 74
lta_result, 75
lta_subgroup, 77

map_cruz, 82
mixed_group_affiliates, 83

noaa_10km_1986_2020, 84

points_to_line, 85
prep_bootstrap_datasets, 86
process_km, 87
process_polygon, 88
process_sightings, 89
process_strata, 90
process_subgroups, 91
process_surveys, 92

read_polygon, 99
region_das, 100
region_segments, 100

segmentize, 101
ships, 103
species_codes, 104
species_translator, 104
strata_area, 105
strata_ccs, 107
strata_cnp, 107
strata_etp, 108
strata_explore, 108
strata_segments, 109
strata_select, 110
subgroup_edits, 110
subgroup_events, 111
subgroup_explorer, 112
subgroup_phases, 112
subgroup_populations, 113
subgroup_sightings, 114
subgroup_subgroups, 114
summarize_bft, 115
summarize_distances, 116
summarize_effort, 116
summarize_sightings, 117
summarize_species, 118