Package 'Tivy'

July 23, 2025

Type Package

Title Toolkit for Investigation and Visualization of Young Anchovies

Version 0.1.1

Description Specialized toolkit for processing biological and fisheries data from Peru's anchovy (Engraulis ringens) fishery. Provides functions to analyze fishing logbooks, calculate biological indicators (length-weight relationships, juvenile percentages), generate spatial fishing indicators, and visualize regulatory measures from Peru's Ministry of Production. Features automated data processing from multiple file formats, coordinate validation, spatial analysis of fishing zones, and tools for analyzing fishing closure announcements and regulatory compliance. Includes built-in datasets of Peruvian coastal coordinates and parallel lines for analyzing fishing activities within regulatory zones.

URL https://github.com/HansTtito/Tivy

BugReports https://github.com/HansTtito/Tivy/issues

Depends R (>= 4.0.0)

Imports dplyr (>= 1.0.0), tidyr (>= 1.0.0), lubridate (>= 1.7.0), stringr (>= 1.4.0), stringi (>= 1.4.0), ggplot2 (>= 3.3.0), leaflet (>= 2.0.0), RColorBrewer (>= 1.1.0), rlang (>= 0.4.0), httr (>= 1.4.0), rvest (>= 1.0.0), jsonlite (>= 1.6.0), pdftools (>= 3.0.0), stats, utils, future (>= 1.21.0), future.apply (>= 1.7.0), patchwork (>= 1.1.0), scales (>= 1.1.0)

Suggests rnaturalearth (>= 0.1.0), rnaturalearthdata, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Language en-US

NeedsCompilation no

Author Hans Ttito [aut, cre] (ORCID: <https://orcid.org/0000-0003-3732-9419>)

Maintainer Hans Ttito <kvttitos@gmail.com>

Repository CRAN

Date/Publication 2025-07-23 08:00:08 UTC

Contents

add_variables *Add variables for juveniles, sample length, distance to coast, and distance category*

Description

Adds new variables to a dataset, including the proportion of juveniles, the total number of individuals in the sample, the distance to the coast, and the distance category.

Usage

```
add_variables(
  data,
  JuvLim = 12,
  distance_type = "haversine",
  window = 0.5,
  unit = "nm",
  coastline = NULL,
  suppress_warnings = TRUE
)
```

Arguments

data	Data frame that must contain latitude (lat_initial) and longitude (lon_initial) coordinates, as well as columns with individual length.
JuvLim	Length threshold to consider juveniles.
distance_type	Type of distance calculation to the coast.
window	Window parameter to smooth the coastline.
unit	Distance unit used in the calculation ("nm", "km", etc.).
coastline	Data frame with coastline coordinates. Must have columns named Long and Lat. If NULL, uses internal dataset peru_coastline.
suppress_warnings	Logical. If TRUE, warnings are suppressed.

Value

Data frame with new variables: juv (proportion of juveniles), sample (total individuals), dc (distance to coast), and dc_cat (categorical distance).

Examples

```
## Not run:
data_hauls <- process_hauls(data_hauls = calas_bitacora)
data_fishing_trips <- process_fishing_trips(data_fishing_trips = faenas_bitacora)
hauls_length <- process_length(data_length = tallas_bitacora)
```

```
data_length_trips <- merge(
  x = data_fishing_trips,
  y = hauls_length,
  by = 'fishing_trip_code'
)
data_total <- merge_length_fishing_trips_hauls(
  data_hauls = data_hauls,
  data_length_fishing_trips = data_length_trips
)

results <- add_variables(data = data_total)

## End(Not run)
```

`apply_catch_weighting` *Apply catch weighting to data frame*

Description

Applies catch weighting to length frequency columns in a data frame. Supports parallel processing for large datasets.

Usage

```
apply_catch_weighting(
  data,
  length_cols,
  catch_col,
  a,
  b,
  parallel = FALSE,
  num_cores = NULL,
  block_size = 10000,
  silence_warnings = TRUE
)
```

Arguments

`data`	Data frame containing length columns and catch data.
`length_cols`	Character vector of length column names.
`catch_col`	Name of the catch column.
`a`	Coefficient of the length-weight relationship.
`b`	Exponent of the length-weight relationship.
`parallel`	Logical. Use parallel processing.
`num_cores`	Number of cores for parallel processing. If NULL, auto-detect.

`block_size` Block size for parallel processing.

`silence_warnings` Logical. Suppress warnings.

Value

Data frame with original columns plus weighted length columns (prefixed with "weighted_").

Examples

```
## Not run:
length_cols <- c("8", "8.5", "9", "9.5", "10", "10.5", "11", "11.5")
weighted_data <- apply_catch_weighting(
  data = fishery_data,
  length_cols = length_cols,
  catch_col = "total_catch",
  a = 0.0001,
  b = 2.984
)

## End(Not run)
```

---

calas_bitacora *Fishing hauls*

Description

Dataset containing information about fishing hauls in the Peruvian sea. Hauls are specific locations where fishing activities are conducted.

Usage

```
calas_bitacora
```

Format

A data.frame with sample data of hauls made by vessels in the Peruvian sea.

Source

Randomly generated data.

Examples

```
data(calas_bitacora)
```

`calculate_distances_vectorized` *Calculate vectorized distances to coastline*

Description

Internal function to calculate distances between multiple points and a coastline. Supports different distance calculation methods and spatial filtering.

Usage

```
calculate_distances_vectorized(
  lon_point,
  lat_point,
  coast_lon,
  coast_lat,
  distance_type,
  window,
  unit,
  resolution = 0.25
)
```

Arguments

`lon_point`	Vector of point longitudes.
`lat_point`	Vector of point latitudes.
`coast_lon`	Vector of coastline longitudes.
`coast_lat`	Vector of coastline latitudes.
`distance_type`	Distance calculation method.
`window`	Spatial filter window in degrees.
`unit`	Distance unit.
`resolution`	Grid resolution (only used if distance_type = "grid").

Value

List with distance vectors and corresponding indices.

calculate_fish_weight *Calculate fish weight from length*

Description

Estimates individual fish weight from length using the length-weight relationship: W = a * L^b

Usage

```
calculate_fish_weight(length, a, b)
```

Arguments

length	Numeric vector of fish lengths.
a	Coefficient of the length-weight relationship.
b	Exponent of the length-weight relationship.

Details

The length-weight relationship follows the allometric equation W = a * L^b, where W is weight, L is length, and a and b are species-specific parameters.

Value

Numeric vector of estimated weights.

References

Froese, R. (2006). Cube law, condition factor and weight–length relationships: history, meta-analysis and recommendations. Journal of Applied Ichthyology, 22(4), 241-253.

calculate_juvenile_percentage *Calculate juvenile percentage*

Description

Calculates the percentage of individuals considered juveniles based on a length threshold.

Usage

```
calculate_juvenile_percentage(
  frequency,
  length,
  juvenile_limit = 12,
  silence_warnings = FALSE
)
```

Arguments

frequency	Numeric vector of length frequencies.
length	Numeric vector of corresponding lengths.
juvenile_limit	Length threshold for juvenile classification.
silence_warnings	Logical. Suppress warnings.

Details

Juvenile percentage calculations are essential for fisheries management decisions, particularly in determining fishing quotas and closure periods.

Value

Percentage of juveniles in the sample.

References

IMARPE (2020). Protocolo Elaboración de la Tabla de Decisión para la determinación del Límite Máximo Total Permisible por temporada de pesca en la pesquería del stock norte-centro de la anchoveta peruana. IMP-DGIRP/AFDPERP, Edición: 05, Revisión 00, 40p.

---

calculate_juvenile_statistics

Calculate juvenile statistics for a group

Description

Helper function that calculates juvenile percentages in number and weight. Used internally by summarize_juveniles_by_group.

Usage

```
calculate_juvenile_statistics(
  frequencies,
  length_values,
  juvenile_limit = 12,
  a = 0.0012,
  b = 3.1242
)
```

Arguments

frequencies	Numeric vector of frequencies by length.
length_values	Numeric vector of corresponding lengths.
juvenile_limit	Length threshold for juveniles.
a	Coefficient of length-weight relationship.
b	Exponent of length-weight relationship.

Value

Data frame with juvenile statistics.

Examples

```
frequencies <- c(10, 15, 25, 30, 20, 10)
lengths <- c(8, 9, 10, 11, 12, 13)
stats <- calculate_juvenile_statistics(frequencies, lengths)
```

---

coast_distance *Vectorized distance to coast*

Description

Estimates the distance between a set of points (lon, lat) and a coastline defined by coordinates. Can be executed sequentially or in parallel, and also return the indices of the nearest coastal points.

Usage

```
coast_distance(
  lon,
  lat,
  coastline = NULL,
  return_indices = FALSE,
  distance_type = "haversine",
  unit = "nm",
  window = 1,
  parallel = FALSE,
  cores = 4
)
```

Arguments

lon	Numeric vector with the longitudes of the points of interest.
lat	Numeric vector with the latitudes of the points of interest.
coastline	Data frame with coastline coordinates. Must have columns named Long and Lat. If NULL, uses internal dataset peru_coastline.
return_indices	Logical. If TRUE, also returns the indices of the nearest coastline points.
distance_type	Type of geographic distance to use: "haversine", "euclidean", "grid".
unit	Unit of measurement for distance: "nm" (nautical miles), "km", etc.
window	Search window in degrees around the point to limit calculations and improve efficiency.
parallel	Logical. If TRUE, performs the calculation in parallel using multiple cores.
cores	Number of cores to use for parallel processing.

Value

If `return_indices = FALSE`, returns a numeric vector with distances to the coast for each point. If `return_indices = TRUE`, returns a list with distance and index components.

Examples

```
## Not run:
data_hauls <- process_hauls(data_hauls = calas_bitacora)
distances <- coast_distance(
  lon = data_hauls$lon_final,
  lat = data_hauls$lat_final,
  distance_type = "haversine",
  unit = "nm",
  parallel = TRUE,
  cores = 2
)

## End(Not run)
```

`convert_numbers_to_weight` *Convert numbers to weight*

Description

Converts numerical length frequencies to weight estimates using length-weight relationship.

Usage

```
convert_numbers_to_weight(data, length_cols, a, b)
```

Arguments

`data`	Data frame with length frequency columns.
`length_cols`	Vector of length column names or numeric values.
`a`	Coefficient of the length-weight relationship.
`b`	Exponent of the length-weight relationship.

Value

Data frame with original columns plus weight columns (prefixed with "weight_").

Examples

```
## Not run:
weight_data <- convert_numbers_to_weight(
  data = frequency_data,
  length_cols = c("8", "8.5", "9", "9.5", "10"),
  a = 0.0012,
  b = 3.1242
)

## End(Not run)
```

`convert_to_date` *Convert dates to standard format*

Description

Converts a vector of dates in various formats to a standard date format. The function tries to parse each date using predefined formats and returns the first valid date found for each entry. If a date cannot be interpreted, it is assigned as `NA`.

Usage

```
convert_to_date(date_vector, output_type = c("date", "datetime"))
```

Arguments

`date_vector`	A character vector containing dates in various formats.
`output_type`	Type of object to return: "date" for Date, "datetime" for POSIXct.

Value

A vector of `Date` or `POSIXct` objects, or `NA` if the date cannot be converted.

Examples

```
dates <- c("2025-04-10", "10/04/2025", "April 10, 2025")
converted_dates <- convert_to_date(dates)
print(converted_dates)
```

`create_fishery_dashboard` *Create fishery dashboard*

Description

Generates a comprehensive dashboard for fishery analysis including juvenile analysis, catch trends, spatial distribution, and summary statistics. This function consolidates all dashboard functionality.

Usage

```
create_fishery_dashboard(
  data,
  date_col = NULL,
  length_cols = NULL,
  a = 1e-04,
  b = 2.984,
  latitude_col = NULL,
  longitude_col = NULL,
  catch_col = NULL,
  juvenile_col = NULL,
  map_xlim = c(-85, -70),
  map_ylim = c(-20, 0),
  color_palette = NULL,
  date_breaks = NULL,
  sort_comparison = FALSE,
  comparison_title = NULL,
  catch_title = NULL,
  map_title = NULL,
  trend_title = NULL
)
```

Arguments

`data`	Data frame with complete fishery data.
`date_col`	Date column name. If NULL, auto-detect.
`length_cols`	Length frequency column names. If NULL, auto-detect.
`a`	Length-weight coefficient.
`b`	Length-weight exponent.
`latitude_col`	Latitude column name. If NULL, auto-detect.
`longitude_col`	Longitude column name. If NULL, auto-detect.
`catch_col`	Catch column name. If NULL, auto-detect.
`juvenile_col`	Juvenile percentage column name. If NULL, auto-detect.
`map_xlim`	Map longitude limits.

map_ylim	Map latitude limits.
color_palette	Custom color palette.
date_breaks	Date axis breaks.
sort_comparison	Sort comparison plot.
comparison_title	Comparison plot title.
catch_title	Catch plot title.
map_title	Map plot title.
trend_title	Trend plot title.

Value

List with individual plots and combined dashboard (if patchwork available).

Examples

```
## Not run:
dashboard <- create_fishery_dashboard(
  data = complete_fishery_data,
  date_col = "date",
  length_cols = paste0("length_", seq(8, 15, 0.5)),
  catch_col = "total_catch",
  latitude_col = "latitude",
  longitude_col = "longitude"
)

dashboard$comparison
dashboard$catch_trends
dashboard$spatial_map
dashboard$trends
dashboard$dashboard

## End(Not run)
```

---

dms_to_decimal *Convert latitude or longitude to decimal degrees*

Description

Converts coordinates expressed in degrees, minutes and seconds (DMS) or degrees and minutes (DM) format to decimal degrees. By default, coordinates are assumed to be in the southern hemisphere (negative latitudes). The function can automatically correct common errors such as minutes or seconds greater than 60.

Usage

```
dms_to_decimal(coordinates, hemisphere = "S", correct_errors = TRUE)
```

Arguments

coordinates	Character vector. Each element should be in formats such as: "D M S", "D M", "17 26 S"
hemisphere	Character indicating hemisphere when not specified in the coordinate. One of "N", "S", "E", "W" or "O". "S" and "W"/"O" generate negative values.
correct_errors	Logical. If TRUE, automatically corrects out-of-range values.

Value

Numeric vector with coordinates converted to decimal degrees.

Examples

```
dms_to_decimal(c("73 15 0"), hemisphere = "W")
```

---

extract_numeric_values

Extract length values from column names

Description

Helper function to extract numerical length values from column names. Handles different naming patterns like "length_8.5", "weighted_9", "8", etc. Uses multiple extraction strategies to ensure robust parsing of column names.

Usage

```
extract_numeric_values(
  column_names,
  use_fallback = TRUE,
  fallback_type = "sequential",
  verbose = FALSE
)
```

Arguments

column_names	Character vector of column names.
use_fallback	Logical. If TRUE, uses fallback strategy when numeric values cannot be extracted.
fallback_type	Character. Type of fallback to use when use_fallback = TRUE. Options: "sequential", "ones", "zeros".
verbose	Logical. Print information about extraction strategy used.

Details

The function uses the following extraction strategies in order:

1. Specific prefixes: "length_", "weighted_", "pond_" followed by numbers
2. Purely numeric column names
3. General pattern: extracts first number found in each name
4. Fallback: uses specified fallback strategy if previous methods fail

Value

Numeric vector of length values extracted from column names.

Examples

```
extract_numeric_values(c("length_8.5", "weighted_10", "pond_12"))
extract_numeric_values(c("8", "10.5", "12"))
extract_numeric_values(c("size_8", "data_10.5", "value_12"))
extract_numeric_values(c("length_8", "no_numbers"), use_fallback = FALSE)
extract_numeric_values(c("bad_name1", "bad_name2"), fallback_type = "ones")
```

---

extract_pdf_data *Extract data from PDF announcements*

Description

Processes PDF files containing official fishing announcements and extracts relevant information such as dates, coordinates, and nautical miles. Handles both local files and URLs.

Usage

```
extract_pdf_data(
  pdf_sources = NULL,
  temp_dir = NULL,
  verbose = TRUE,
  max_retries = 3
)
```

Arguments

pdf_sources	Character vector of PDF file paths or URLs.
temp_dir	Temporary directory for downloaded files. If NULL, uses tempdir().
verbose	Show processing messages.
max_retries	Maximum download retries for URLs.

Value

Data frame with extracted announcement information including coordinates, dates, and nautical mile distances.

Examples

```
## Not run:
pdf_files <- c("announcement1.pdf", "announcement2.pdf")
results <- extract_pdf_data(pdf_sources = pdf_files)

pdf_urls <- c(
  "https://example.com/announcement1.pdf",
  "https://example.com/announcement2.pdf"
)
results <- extract_pdf_data(pdf_sources = pdf_urls)

## End(Not run)
```

---

faenas_bitacora *Fishing trips*

Description

Dataset containing information about fishing trips conducted along the Peruvian littoral.

Usage

```
faenas_bitacora
```

Format

A data.frame with sample data of fishing trips made by vessels in the Peruvian sea.

Source

Randomly generated data.

Examples

```
data(faenas_bitacora)
```

`fetch_fishing_announcements` *Fetch fishing announcements from external sources*

Description

Retrieves fishing announcements from official websites within a specified date range. This function is specifically designed for PRODUCE (Peru) but can be adapted for other sources.

Usage

```
fetch_fishing_announcements(
  start_date,
  end_date,
  download = FALSE,
  download_dir = "downloads",
  batch_size = 10,
  verbose = TRUE,
  source_url = NULL,
  max_records = 5000
)
```

Arguments

`start_date`	Start date in "dd/mm/yyyy" format.
`end_date`	End date in "dd/mm/yyyy" format.
`download`	Logical. Download PDF files.
`download_dir`	Directory for downloaded files.
`batch_size`	Records per request.
`verbose`	Print detailed information.
`source_url`	Base URL for the announcement source. Defaults to the PRODUCE page: https://consultasenlinea.produce.gob.pe/ConsultasEnLinea/consultas.web/comunicados/suspensionPreventiva
`max_records`	Maximum records to retrieve.

Value

Data frame with announcement information and download links.

Examples

```
## Not run:
announcements <- fetch_fishing_announcements(
  start_date = "01/01/2023",
  end_date = "31/12/2023"
```

```
)

announcements <- fetch_fishing_announcements(
  start_date = "01/01/2023",
  end_date = "31/01/2023",
  download = TRUE,
  download_dir = "announcements"
)

## End(Not run)
```

`find_column` *Find column by pattern matching*

Description

Searches for a column in a data frame using multiple pattern options. If multiple columns match, warns the user and returns the first match.

Usage

```
find_column(patterns, column_names, verbose = FALSE)
```

Arguments

`patterns`	Character vector of regex patterns to search for.
`column_names`	Character vector of column names to search in.
`verbose`	Logical. If TRUE, prints detailed matching information.

Value

Integer index of the first matching column, or NULL if no match found.

Examples

```
cols <- c("codigo_faena", "numero_cala", "especie", "especies_capturadas")
species_patterns <- c("especie", "species", "sp")
col_index <- find_column(species_patterns, cols)
```

`find_columns_by_pattern` *Find columns by pattern*

Description

Identifies columns in a data frame that match a specific pattern. Useful for finding length columns, weight columns, etc.

Usage

```
find_columns_by_pattern(data, pattern = "weighted_", sort = TRUE)
```

Arguments

`data`	Data frame to search.
`pattern`	Regular expression pattern to match.
`sort`	Logical. Sort results numerically.

Value

Character vector of matching column names.

Examples

```
# Create dummy data
data <- data.frame(
  weighted_8.5 = c(1, 2, 3),
  weighted_10 = c(4, 5, 6),
  `8` = c(7, 8, 9),
  other = c(10, 11, 12)
)

# Find weighted columns
weighted_cols <- find_columns_by_pattern(data, pattern = "weighted_")

# Find numeric-only named columns (e.g., "8")
length_cols <- find_columns_by_pattern(data, pattern = "")
```

`format_extracted_data` *Format extracted announcement data*

Description

Formats and filters data extracted from announcements, converting dates to proper formats and allowing filtering by date range.

Usage

```
format_extracted_data(
  data,
  min_date = NULL,
  max_date = NULL,
  convert_coordinates = TRUE
)
```

Arguments

`data`	Data frame with structure from extract_pdf_data.
`min_date`	Minimum date for filtering (YYYY-MM-DD format or Date/POSIXct object).
`max_date`	Maximum date for filtering (YYYY-MM-DD format or Date/POSIXct object).
`convert_coordinates`	Logical. Convert DMS coordinates to decimal.

Value

Data frame with formatted and filtered announcement data.

Examples

```
## Not run:
formatted_data <- format_extracted_data(raw_data)

filtered_data <- format_extracted_data(
  data = raw_data,
  min_date = "2024-11-01",
  max_date = "2024-12-31"
)

## End(Not run)
```

get_length_range *Get length range from frequencies*

Description

Finds the minimum or maximum length with positive frequency.

Usage

```
get_length_range(frequency, length, type = "min")
```

Arguments

frequency	Numeric vector of length frequencies.
length	Numeric vector of corresponding lengths.
type	Either "min" or "max" to specify which range to return.

Value

Minimum or maximum length value with frequency > 0.

Examples

```
freq <- c(0, 0, 1, 2, 3, 4, 2, 1, 0)
lengths <- c(5, 6, 7, 8, 9, 10, 11, 12, 13)
min_length <- get_length_range(freq, lengths, type = "min")
max_length <- get_length_range(freq, lengths, type = "max")
```

land_points *Points on land*

Description

Classifies a set of geographic coordinates (longitude and latitude) as "land" or "sea" according to their relative position to a coastline. A point is considered to be on land if its longitude is greater than that of its nearest point on the coastline.

Usage

```
land_points(
  x_point,
  y_point,
  coastline = NULL,
  parallel = FALSE,
  cores = 4,
  distance_type = "haversine",
  window = 0.5,
  unit = "nm"
)
```

Arguments

x_point	Numeric vector of longitudes (in decimal degrees).
y_point	Numeric vector of latitudes (in decimal degrees).
coastline	Data frame with coastline coordinates. Must have columns named Long and Lat. If NULL, uses internal dataset peru_coastline.
parallel	Logical. If TRUE, performs the calculation in parallel using multiple cores.
cores	Number of cores to use for parallel processing.
distance_type	Type of geodesic distance to use in the calculation.
window	Geographic window in degrees to reduce the number of coastline points to consider.
unit	Unit of measurement for distance: "km" or "nm".

Value

Text vector of the same length as x_point, indicating whether each point is on "land" or "sea". NA values are maintained as NA.

Examples

```
## Not run:
data_hauls <- process_hauls(data_hauls = calas_bitacora)
result <- land_points(
  x_point = data_hauls$lon_final,
  y_point = data_hauls$lat_final
)
table(result)

## End(Not run)
```

merge_length_fishing_trips_hauls *Merge fishing trips, length and hauls data*

Description

Joins data from fishing trips, length and hauls, combining catches by species, length ranges (minimum and maximum) and spatial-temporal information of each haul.

Usage

```
merge_length_fishing_trips_hauls(data_hauls, data_length_fishing_trips)
```

Arguments

data_hauls	Data frame processed with process_hauls().
data_length_fishing_trips	Data frame with length data by fishing trip and haul.

Value

Data frame with consolidated data from fishing trips, length and hauls.

Examples

```
## Not run:
data_hauls <- process_hauls(data_hauls = calas_bitacora)
data_fishing_trips <- process_fishing_trips(data_fishing_trips = faenas_bitacora)
data_length <- process_length(data_length = tallas_bitacora)

data_length_fishing_trips <- merge(
   x = data_length,
   y = data_fishing_trips,
   by = "fishing_trip_code",
   all = TRUE
)

data_total <- merge_length_fishing_trips_hauls(
   data_hauls = data_hauls,
   data_length_fishing_trips = data_length_fishing_trips
)

## End(Not run)
```

peru_coastline *Peruvian coastline*

Description

Dataset containing the coastline of Peru represented as a spatial object.

Usage

```
peru_coastline
```

Format

A data.frame with the following fields:

Long Longitude of the Peruvian coastline

Lat Latitude of the Peruvian coastline

Source

Marine Institute of Peru (IMARPE)

Examples

```
data(peru_coastline)
```

peru_coast_parallels *Lines parallel to the Peruvian coast*

Description

Dataset containing lines parallel to the Peruvian coast at different distances. These lines are useful for spatial analyses related to fishery management.

Usage

```
peru_coast_parallels
```

Format

A list of data.frames, each representing a line parallel to the Peruvian coast.
Each data frame contains:

lon Longitude corresponding to the isoparalittoral area

lat Latitude corresponding to the isoparalittoral area

dc Distance to coast category (10-200)

Source

Indicate the data source

Examples

```
data(peru_coast_parallels)
```

`peru_isoparalitoral_areas`
Isoparalittoral areas

Description

Dataset containing isoparalittoral areas of the Peruvian coast. These areas represent zones with similar characteristics along the littoral.

Usage

```
peru_isoparalitoral_areas
```

Format

A data.frame with the following attributes:

lon Longitude corresponding to the isoparalittoral area

lat Latitude corresponding to the isoparalittoral area

area Isoparalittoral Area code

grad Latitude category every 0.5 degrees (3 - 19.5)

dc Distance to coast category (10-200)

Source

Marine Institute of Peru (IMARPE)

Examples

```
data(peru_isoparalitoral_areas)
```

`plot_fishing_zones` *Plot fishing zones*

Description

Creates visualizations of fishing zones using either ggplot2 (static) or leaflet (interactive). This function consolidates all zone plotting functionality.

Usage

```
plot_fishing_zones(
  data,
  coastline = NULL,
  parallels = NULL,
  type = "static",
  title = NULL,
  colors = NULL,
  show_legend = FALSE,
  legend_title = NULL,
  zone_labels = NULL,
  add_grid = FALSE,
  base_layers = FALSE,
  minimap = FALSE
)
```

Arguments

`data`	Data frame with fishing zone coordinates and metadata.
`coastline`	Data frame with coastline coordinates (columns "Long" and "Lat"). If NULL, uses internal dataset.
`parallels`	List of data frames with coast-parallel lines.
`type`	Plot type: "static" for ggplot2 or "interactive" for leaflet.
`title`	Plot title.
`colors`	Vector of colors for zones. If NULL, auto-generated.
`show_legend`	Logical. Show legend/layer control.
`legend_title`	Legend title.
`zone_labels`	Vector of custom labels for zones.
`add_grid`	Logical. Add coordinate grid (static only).
`base_layers`	Logical. Include multiple base map layers (interactive only).
`minimap`	Logical. Add minimap (interactive only).

Value

ggplot object (static) or leaflet object (interactive).

Examples

```
## Not run:
plot_fishing_zones(
  data = zone_data,
  coastline = coastline_data,
  type = "static",
  title = "Fishing Zones",
  show_legend = TRUE
)

plot_fishing_zones(
  data = zone_data,
  coastline = coastline_data,
  type = "interactive",
  base_layers = TRUE,
  minimap = TRUE
)

## End(Not run)
```

---

plot_juvenile_analysis

Plot juvenile analysis

Description

Creates comprehensive visualizations for juvenile fish analysis including bar charts, line plots, and comparative analyses. This function consolidates all juvenile plotting functionality.

Usage

```
plot_juvenile_analysis(
  data,
  x_var,
  fill_var = NULL,
  length_cols = NULL,
  a = 0.0012,
  b = 3.1242,
  x_date_breaks = NULL,
  plot_type = "bars",
  title = NULL,
  subtitle = NULL,
  sort_by = "x",
  color_palette = NULL,
  facet_var = NULL,
  facet_cols = 2,
  bar_position = "dodge",
```

```
  y_limits = c(0, 100),
  use_facet_wrap = TRUE,
  group_by_type = TRUE,
  reference_line = NULL,
  theme_style = "light",
  legend_position = "bottom",
  rotate_x_labels = TRUE,
  na_to_zero = FALSE
)
```

Arguments

data	Data frame with juvenile analysis data.
x_var	Column name for x-axis variable.
fill_var	Column name for fill/color variable.
length_cols	Vector of length frequency column names.
a	Length-weight relationship coefficient.
b	Length-weight relationship exponent.
x_date_breaks	Date breaks for x-axis (e.g., "1 day", "1 month").
plot_type	Plot type: "bars", "lines", "points", "mixed".
title	Plot title.
subtitle	Plot subtitle.
sort_by	Sorting method: "x", "number", "weight".
color_palette	Custom color palette.
facet_var	Variable for faceting.
facet_cols	Number of facet columns.
bar_position	Bar position: "dodge", "stack", "fill".
y_limits	Y-axis limits.
use_facet_wrap	Use facet wrap for juvenile type.
group_by_type	Group by juvenile type when not faceting.
reference_line	Reference line value (e.g., legal limit).
theme_style	Theme style: "classic", "minimal", "light", "dark".
legend_position	Legend position.
rotate_x_labels	Rotate x-axis labels.
na_to_zero	Convert NA values to zeros.

Value

ggplot object.

Examples

```
## Not run:
plot_juvenile_analysis(
  data = fishery_data,
  x_var = "date",
  length_cols = paste0("length_", seq(8, 15, 0.5))
)

plot_juvenile_analysis(
  data = fishery_data,
  x_var = "date",
  fill_var = "vessel",
  length_cols = length_columns,
  plot_type = "mixed",
  reference_line = 10,
  title = "Juvenile Analysis by Vessel and Date"
)

## End(Not run)
```

`process_fishing_trips` *Process fishing trip data from PRODUCE sitrapesca files*

Description

Processes fishing trip data from PRODUCE logbooks. Automatically detects required columns and creates standardized output with proper date conversion.

Usage

```
process_fishing_trips(data_fishing_trips, verbose = FALSE)
```

Arguments

`data_fishing_trips`	Data frame with raw fishing trip data.
`verbose`	Logical. Print column mapping information.

Value

Data frame with 6 standardized columns including trip code, vessel information, and trip dates.

Examples

```
fishing_trips <- process_fishing_trips(data_fishing_trips = faenas_bitacora)
fishing_trips <- process_fishing_trips(data_fishing_trips = faenas_bitacora, verbose = TRUE)
```

process_hauls *Process fishing haul data from PRODUCE sitrapesca files*

Description

Processes fishing haul data from PRODUCE logbooks. Automatically detects required columns and creates standardized output with coordinates converted to decimal degrees.

Usage

```
process_hauls(data_hauls, correct_coordinates = TRUE, verbose = FALSE)
```

Arguments

data_hauls	Data frame with raw haul data.
correct_coordinates	Logical. Correct coordinate errors during conversion.
verbose	Logical. Print column mapping information.

Value

Data frame with 16 standardized columns including fishing trip code, haul number, dates, coordinates, species, and catch data.

Examples

```
processed_hauls <- process_hauls(data_hauls = calas_bitacora)
processed_hauls <- process_hauls(data_hauls = calas_bitacora, verbose = TRUE)
```

process_length *Process length data from hauls*

Description

Processes length data from PRODUCE logbooks. Automatically detects required columns and transforms from long to wide format.

Usage

```
process_length(data_length, verbose = FALSE)
```

Arguments

data_length	Data frame with raw length data.
verbose	Logical. Print column mapping information.

Value

Data frame with length by haul in wide format with individual columns for each length class.

Examples

```
length_data <- process_length(data_length = tallas_bitacora)
length_data <- process_length(data_length = tallas_bitacora, verbose = TRUE)
```

`summarize_juveniles_by_group` *Summarize juveniles by group*

Description

Calculates juvenile percentages by specified groups, both in number and weight. Uses modern dplyr approach for efficient processing. Can auto-detect length columns if not specified.

Usage

```
summarize_juveniles_by_group(
  data,
  group_cols,
  length_cols = NULL,
  juvenile_limit = 12,
  a = 0.0012,
  b = 3.1242,
  remove_empty = TRUE,
  verbose = FALSE
)
```

Arguments

`data`	Data frame with length frequency data.
`group_cols`	Vector of column names for grouping.
`length_cols`	Vector of length column names or indices. If NULL, auto-detection is attempted.
`juvenile_limit`	Length threshold for juveniles.
`a`	Coefficient of length-weight relationship.
`b`	Exponent of length-weight relationship.
`remove_empty`	Logical. Remove groups with no data.
`verbose`	Logical. Print information about detected columns.

Value

Data frame with juvenile statistics by group.

Examples

```
## Not run:
juvenile_summary <- summarize_juveniles_by_group(
  data = fishery_data,
  group_cols = "date",
  juvenile_limit = 12
)

## End(Not run)
```

---

tallas_bitacora *Length data sampled from hauls made by vessels in the Peruvian sea.*

Description

Dataset containing information about lengths of marine species captured along the Peruvian littoral.

Usage

```
tallas_bitacora
```

Format

A data.frame with sample data of lengths sampled from hauls made by vessels in the Peruvian sea.

Source

Indicate the data source

Examples

```
data(tallas_bitacora)
```

---

validate_fishing_trip_data *Validate processed fishing trip data quality*

Description

Validates data quality metrics for processed fishing trip data.

Usage

```
validate_fishing_trip_data(processed_trips)
```

Arguments

`processed_trips`
: Data frame returned by process_fishing_trips().

Value

List with data quality metrics including completeness scores and issue counts.

`validate_haul_data` *Validate processed haul data quality*

Description

Validates data quality metrics for processed haul data.

Usage

```
validate_haul_data(processed_hauls)
```

Arguments

`processed_hauls`
: Data frame returned by process_hauls().

Value

List with data quality metrics including completeness scores and issue counts.

`validate_length_data` *Validate processed length data quality*

Description

Validates data quality metrics for processed length data.

Usage

```
validate_length_data(processed_length)
```

Arguments

`processed_length`
: Data frame returned by process_length().

Value

List with data quality metrics including completeness scores and distribution statistics.

`validate_polygon_data` *Validate data for polygon creation*

Description

Validates that a data frame contains the necessary columns for creating fishing zone polygons. Checks for either coordinate-based or distance-based polygon definition data.

Usage

```
validate_polygon_data(data)
```

Arguments

`data`	Data frame to validate. Must contain polygon definition columns.

Details

The function requires either:

- Coordinate-based: StartLatitude, EndLatitude, StartLongitude, EndLongitude
- Distance-based: StartLatitude, EndLatitude, StartNauticalMiles, EndNauticalMiles

Value

Returns TRUE invisibly if validation passes, otherwise throws an error.

Examples

```
# Coordinate-based polygon data
coord_data <- data.frame(
  StartLatitude = "15 30 S",
  EndLatitude = "15 45 S",
  StartLongitude = "75 30 W",
  EndLongitude = "75 45 W"
)
validate_polygon_data(coord_data)

# Distance-based polygon data
distance_data <- data.frame(
  StartLatitude = "15 30 S",
  EndLatitude = "15 45 S",
  StartNauticalMiles = 5,
  EndNauticalMiles = 15
)
validate_polygon_data(distance_data)
```

`weight_by_catch` *Weight length frequencies by total catch*

Description

Scales observed length frequencies based on total recorded catch using length-weight relationship.

Usage

```
weight_by_catch(frequency, catch, length, a, b, silence_warnings = FALSE)
```

Arguments

`frequency`	Numeric vector of observed length frequencies.
`catch`	Total catch amount (in kg or tons).
`length`	Numeric vector of lengths corresponding to frequencies.
`a`	Coefficient of the length-weight relationship.
`b`	Exponent of the length-weight relationship.
`silence_warnings`	Logical. Suppress warning messages.

Details

Catch weighting is used to estimate the size composition of catches when only sub-samples are measured for length frequency analysis.

Value

Numeric vector of weighted frequencies.

References

IMARPE (2020). Protocolo Elaboración de la Tabla de Decisión para la determinación del Límite Máximo Total Permisible por temporada de pesca en la pesquería del stock norte-centro de la anchoveta peruana. IMP-DGIRP/AFDPERP, Edición: 05, Revisión 00, 40p.

Index