Package 'FishDiveR'

November 7, 2026

Title Classify Aquatic Animal Behaviours from Vertical Movement Data

Version 1.3.0

Maintainer Calvin Beale <calvin.beale.8@gmail.com>

Description Quantitatively analyse depth time-series data from pop-up satellite archival tags (PSATs) through the application of continuous wavelet transformation (CWT) combined with Principal Component Analysis (PCA), and k-means clustering. Import, crop, and plot depth time-depth records (TDRs). Using CWT to detect important signals within the non-stationary data, we create daily wavelet statistics to summarise vertical movements on different wavelet periods and combine with daily and diel depth statistics. Classify depth time-series with unsupervised k-means clustering into 24-hour periods of vertical movement behaviour with distinct patterns of vertical movement. Plot example days from each behaviour cluster, and plot the TDR coloured by cluster. Based on principles of combining CWT with k-means first developed by Sakamoto et al. (2009) <doi:10.1371/journal.pone.0005379> and redeveloped by Beale et al. (2026) <doi:10.1186/s40462-025-00622-w>.

License GPL (>= 3)

URL https://calvinsbeale.github.io/FishDiveR/, https://github.com/calvinsbeale/FishDiveR, https://calvinbeale.co.uk/fishdiver/

BugReports https://github.com/calvinsbeale/FishDiveR/issues

Imports cluster, cowplot, data.table, dplyr, FactoMineR, geometry, ggplot2, gridExtra, lubridate, moments, patchwork, colorspace, rgl, Rfast, rlang, scales, suncalc, tidyr, WaveletComp

Suggests bslib, DT, knitr, rmarkdown, rstudioapi, shiny (>= 1.7.0), tcltk, testthat (>= 3.0.0)

Depends R (>= 3.5.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Calvin Beale [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-3505-8929>)

Repository CRAN

Date/Publication 2026-11-07 12:10:15 UTC

Contents

combine_data 2
create_depth_stats 3
create_wavelet 5
create_wavelet_stats 7
import_tag_data 9
k_clustering 10
pca_data 12
pca_results 13
pca_scores 15
plot_clusters 16
plot_cluster_TDR 18
plot_TDR 20
plot_time_at 22
run_app 23
sampling_report 24
select_k 26
thin_archive 27

Index **29**

`combine_data` *Import depth statistics and combine with PC scores*

Description

This function imports the depth statistics from each of the tags listed in tag_vector, and outputs a combined data frame then combines the depth statistics from each tag with the principal component scores, and outputs a data frame with the appropriate unique_tag_ID if necessary, ready for use in k-means clustering.

Usage

```
combine_data(
  tag_vector = tag_list,
  data_folder = NULL,
  pc_scores = scores,
  output = FALSE,
  output_folder = NULL,
```

```
  verbose = FALSE
)
```

Arguments

tag_vector	A character vector of tag IDs. E.g. 'c("123456", "456283", "AB98XJ").
data_folder	Parent folder path with separate folders for each tag data. E.g. "C:/Tag data". Defaults to 'data_dir'
pc_scores	Data frame of principal component scores extracted through PCA on wavelet statistics. Output of 'pca_scores()' function.
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Value

A data frame containing the combined depth statistics and principal component scores from each of the tags listed in tag_vector

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load pc_results
pc_scores <- readRDS(file.path(filepath, "data/4_PCA/pc_scores.rds"))

# Run combine_data function
combined_stats <- combine_data(
  tag_vector = "data",
  data_folder = filepath,
  pc_scores = pc_scores,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

---

create_depth_stats *Create depth statistics*

Description

create_depth_stats creates the various daily and diel depth statistics for each day

Usage

```
create_depth_stats(
  archive,
  tag_ID,
  diel = FALSE,
  sunrise_time = NULL,
  sunset_time = NULL,
  GPS = FALSE,
  sunset_type = "civil",
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

archive	Data frame containing processed time series depth data
tag_ID	Unique tag identification number in a vector of characters. E.g. "123456"
diel	Include diel statistics when TRUE
sunrise_time	Sunrise time (local time zone) in 24-hour clock. E.g. "05:45:00"
sunset_time	Sunset time (local time zone) in 24-hour clock. E.g. "18:30:00"
GPS	Either FALSE or the location of the GPS file containing columns 'date', 'lat' (latitude) and 'lon' (longitude) if one exists. 'date' columns must be in a format readable by lubridate::dmy()
sunset_type	Choose which type of sunset to include 'NULL', 'civil', 'nautical', or 'astronomical'
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Value

A set of statistics calculated daily for the depth data. If diel is 'TRUE', additional diel statistics will be returned. An attribute 'diel' with value 'TRUE' is given when diel statistics are included.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load archive_days
archive_days <- readRDS(file.path(filepath, "data/archive_days.rds"))

# Run create_depth_stats function
depthStats <- create_depth_stats(
  archive = archive_days,
```

```
  tag_ID = "data",
  diel = TRUE,
  sunrise_time = "06:00:00",
  sunset_time = "18:00:00",
  GPS = file.path(filepath, "data/GPS.csv"),
  sunset_type = "civil",
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

`create_wavelet` *Create and plot the wavelet power spectrum*

Description

`create_wavelet` creates the a wavelet spectrum using WaveletComp package. Optionally loads and plots an existing my.w object.

Usage

```
create_wavelet(
  archive,
  tag_ID,
  wv_period_hours = 24,
  sampling_frequency = NULL,
  load_existing_wavelet = FALSE,
  suboctaves = 12,
  lower_period_mins = 5,
  upper_period_hours = 24,
  pval = FALSE,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE,
  plot_wavelet = TRUE,
  max_period_ticks = 10,
  plot_width = 800,
  plot_height = 400,
  interactive_mode = TRUE
)
```

Arguments

`archive`	Data frame containing processed time series depth data
`tag_ID`	Unique tag identification number in a vector of characters. E.g. "123456"

`wv_period_hours`
: Time resolution in hours to calculate wavelet. Currently only supports the default of 24 hours as this package is created to investigate daily diving behaviour. Defaults to 24.

`sampling_frequency`
: Sampling frequency of depth data in seconds. Defaults to time between first and second depth record. Recommended to leave blank.

`load_existing_wavelet`
: Load an existing my.w wavelet object from the output_folder. Defaults to FALSE.

`suboctaves`
: number of suboctaves between each logarithmic period. E.g. between 24 and 12 hours. Highly recommended to use 12, for easy of interpretation of hours and signal present (daily, diel, tidal).

`lower_period_mins`
: Lower period of the wavelet sampling in minutes. Cannot be less than sampling frequency. Defaults to 5 minutes.

`upper_period_hours`
: Upper period of the wavelet sampling in days. Defaults to 24 hours.

`pval`
: Produce p-values or not. True or False. Default set to FALSE, see `WaveletComp::analyze.wavelet()` for further details. P-values not used in further analysis, and increase computation time and file size.

`output`
: Logical. If TRUE, output is saved to `output_folder`. Defaults to FALSE.

`output_folder`
: Output folder path. If `output = TRUE`, `output_folder` must be provided. Defaults to NULL.

`verbose`
: Logical. If TRUE, progress messages are shown. Defaults to FALSE.

`plot_wavelet`
: TRUE or FALSE. Plot the wavelet spectrum and mean power?

`max_period_ticks`
: Number of ticks displayed on the period (y) axis in plots.

`plot_width`
: Width of the wavelet spectrum plot output. Defaults to 800.

`plot_height`
: Height of the wavelet spectrum plot output. Defaults to 400.

`interactive_mode`
: Used for testing the package only. Defaults to TRUE.

Details

Uses `WaveletComp::analyze.wavelet()` to create a univariate wavelet power spectrum for the depth data imported, see `WaveletComp::analyze.wavelet()` for more details. Plots mean wavelet power using `WaveletComp::wt.avg()`. If you have errors allocating large vectors try using library(bigmemory) and create a big matrix with big_mat <- big.matrix(nrow = 1e7, ncol = 10, type = "double") then run your code again. This allows greater range between lower and upper periods

Value

When output = TRUE, returns an object of class "analyze.wavelet" from package 'WaveletComp'. Additionally outputs a plot of the wavelet spectrum, and a plot of the mean power per period.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load archive_days
archive_days <- readRDS(file.path(filepath, "data/archive_days.rds"))

# Run create_wavelet function
my.w <- create_wavelet(
  archive = archive_days,
  tag_ID = "data",
  wv_period_hours = 24,
  sampling_frequency = NULL,
  load_existing_wavelet = FALSE,
  suboctaves = 12,
  lower_period_mins = 30,
  upper_period_hours = 24,
  pval = FALSE,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE,
  plot_wavelet = FALSE,
  max_period_ticks = 10,
  plot_width = 800,
  plot_height = 400,
  interactive_mode = FALSE
)
```

---

create_wavelet_stats *create_wavelet_stats*

Description

create_wavelet_stats aggregates the wavelet variables over the specified time periods

Usage

```
create_wavelet_stats(
  wavelet,
  tag_ID,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

wavelet	An object of class "analyze.wavelet" from package 'WaveletComp'
tag_ID	Unique tag identification number in a vector of characters. E.g. "123456"
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Value

A data frame containing the seven wavelet statistics for each period. One observation is available per period per day:

- Amplitude_mean
- Amplitude_variance
- Mean_sq_power
- Power_mean
- Power_variance
- Phase_mean
- Phase_variance

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load my.w wavelet object
my.w <- readRDS(file.path(filepath, "data/1_Wavelets/data_wavelet.rds"))

# Run create_wavelet_stats function on wavelet object
waveStats <- create_wavelet_stats(
  wavelet = my.w,
  tag_ID = "data",
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

`import_tag_data` *Load time-depth series data from csv file*

Description

`import_tag_data` processes the time-series depth data of marine animal tags. Data to import should be a csv file with a 'date_time' column and a depth column. Data is cropped by deployment and release times.

Usage

```
import_tag_data(
  tag_ID,
  tag_deploy_UTC,
  tag_release_UTC,
  archive,
  date_time_col = 1,
  depth_col = 2,
  temp_col = NA,
  time_zone,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

`tag_ID`	Unique tag identification number in a vector of characters. E.g. "123456"
`tag_deploy_UTC`	UTC deployment time in the allowed `POSIXct` format: E.g. "2013-10-25 02:46:00"
`tag_release_UTC`	UTC release time in the allowed `POSIXct` format: E.g. "2014-04-23 23:17:35"
`archive`	File path of the time-series depth archive. E.g. ("C:/Tag data/123456/123456-Archive.csv")
`date_time_col`	Column number of the date time series
`depth_col`	Column number of the depth series
`temp_col`	(Optional) Column number of temperature series
`time_zone`	Time zone of the data. E.g. "Asia/Tokyo"
`output`	Logical. If TRUE, output is saved to `output_folder`. Defaults to FALSE.
`output_folder`	Output folder path. If `output = TRUE`, `output_folder` must be provided. Defaults to NULL.
`verbose`	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Details

Data are cropped to full days from midnight to midnight in local time based on the time zone supplied. If `output = TRUE`, the cropped data are saved as `archive_days.rds` in the tag's folder within `output_folder`, with the settings used (archive, deployment and release times, columns and time zone) in `import_settings.rds`. The app uses these to warn when a tag's settings have changed since it was imported.

A warning is given if the sampling interval is not constant throughout the cropped data (e.g. gaps from missing records), listing the intervals detected and how many records have each. Use `sampling_report()` on the imported data to see where the irregularities are.

Value

A data frame of processed tag data. Columns kept are:

- 'date' a POSIXct date_time object in format "yyyy-mm-dd hh:mm:ss"
- 'depth' numerical depth data
- 'temp' numerical temperature data
- 'date_only' an as.Date version of the 'date' column
- An attribute 'time_zone' is added to the date frame containing the time zone of the 'date'

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Run import_tag_data function on tag archive csv file
archive_days <- import_tag_data(
  tag_ID = "data",
  tag_deploy_UTC = "2000-01-01 00:00:00",
  tag_release_UTC = "2000-01-11 23:59:00",
  archive = file.path(filepath, "data/data-Archive.csv"),
  date_time_col = 1,
  depth_col = 2,
  temp_col = NA,
  time_zone = "Asia/Tokyo",
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

`k_clustering` *Perform k-means*

Description

`k_clustering` performs k-means clustering on the PC scores with the selected value of k

Usage

```
k_clustering(
  kmeans_data,
  standardise = TRUE,
  k,
  nstart = 50,
  polygon = FALSE,
  output = TRUE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

kmeans_data	Data frame containing the combined PC scores and depth statistics to perform k-means on. Output from the 'combine_data()' function.
standardise	TRUE or FALSE. Whether or not to standardise the data. Defaults to TRUE.
k	Numerical. Value of k to use for analysis.
nstart	Numerical. Value of nstart for k-means analysis.
polygon	TRUE or FALSE. Plot polygons for cluster with more than 3 data points. Defaults to FALSE.
output	TRUE or FALSE. Whether or not to output the results. Defaults to TRUE.
output_folder	Output folder path. If `output = TRUE`, `output_folder` must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Details

This function relies on random initialisation in k-means clustering. For reproducible results, users may wish to set a random seed prior to calling this function using `set.seed()`.

Value

An object of class 'kmeans' containing the k-means clustering data for the data frame. Additionally plots a 3D cluster plot of the top three Principal Components.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load kmeans_data
kmeans_data <- readRDS(file.path(filepath, "data/5_k-means/combined_stats.rds"))


# Full example using the complete dataset.
# Set output to TRUE for real use!
```

```
kmeans_result <- k_clustering(
  kmeans_data = kmeans_data,
  standardise = TRUE,
  k = 4,
  nstart = 50,
  polygon = FALSE,
  output = FALSE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

---

pca_data *Prepare all data for Principal Component Analysis*

Description

pca_data loads the wavelet statistics for each of the tags listed in 'tag_vector'. Performs various checks to ensure compatibility of wavelets, and combines them into a data frame containing only the chosen statistics.

Usage

```
pca_data(
  tag_vector,
  data_folder = data_dir,
  phase_mean = FALSE,
  phase_variance = FALSE,
  power_mean = TRUE,
  power_variance = TRUE,
  mean_sq_power = FALSE,
  amplitude_mean = TRUE,
  amplitude_variance = FALSE,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

tag_vector	A character vector of tag IDs. E.g. 'c("123456", "456283", "AB98XJ").
data_folder	Parent folder path with separate folders for each tag data. E.g. "C:/Tag data". Defaults to 'data_dir'
phase_mean	TRUE or FALSE to include this wavelet statistic. Default FALSE
phase_variance	TRUE or FALSE to include this wavelet statistic. Default FALSE

power_mean	TRUE or FALSE to include this wavelet statistic. Default TRUE
power_variance	TRUE or FALSE to include this wavelet statistic. Default TRUE
mean_sq_power	TRUE or FALSE to include this wavelet statistic. Default FALSE
amplitude_mean	TRUE or FALSE to include this wavelet statistic. Default TRUE
amplitude_variance	TRUE or FALSE to include this wavelet statistic. Default FALSE
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Value

A data frame with the combined data for all tag ID's listed, containing the wavelet statistics to be used in Principal Component Analysis.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Run pca_data function
pc_data <- pca_data(
  tag_vector = c("data"),
  data_folder = filepath,
  phase_mean = FALSE,
  phase_variance = FALSE,
  power_mean = TRUE,
  power_variance = TRUE,
  mean_sq_power = FALSE,
  amplitude_mean = TRUE,
  amplitude_variance = FALSE,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

---

pca_results *Perform Principal Component Analysis*

Description

pca_results performs Principal Component Analysis on the pc_data data frame containing statistics from wavelet analysis

Usage

```
pca_results(
  pc_data,
  standardise = TRUE,
  No_pcs = NULL,
  PCV = NULL,
  plot_eigenvalues = TRUE,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE,
  interactive_mode = TRUE
)
```

Arguments

pc_data	Data frame containing the output of the pca_data() function.
standardise	TRUE or FALSE. Whether or not to standardise the data. Default TRUE.
No_pcs	Numerical. Number of principal components to retain. Null by default
PCV	Numerical. Percentage of cumulative variance to retain. Null by default
plot_eigenvalues	TRUE or FALSE. Plot PC eigenvalues and general loadings. Default TRUE.
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.
interactive_mode	TRUE or FALSE. Used for testing the package. Default FALSE.

Value

A PCA object from 'FactoMineR' package containing the output of the Principal Component Analysis.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load pc_data
pc_data <- readRDS(file.path(filepath, "data/4_PCA/pc_data.rds"))


# Run a minimal, fast pca_results example
pc_results <- pca_results(
  pc_data = pc_data,
  standardise = TRUE,
  No_pcs = 1,
```

```
  PCV = NULL,
  plot_eigenvalues = FALSE,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE,
  interactive_mode = FALSE
)


# Full example using the complete dataset
# Run pca_results function
pc_results <- pca_results(
  pc_data = pc_data,
  standardise = TRUE,
  No_pcs = 3,
  PCV = NULL,
  plot_eigenvalues = TRUE,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE,
  interactive_mode = FALSE
)
```

`pca_scores` *Calculate Principal Component Analysis Scores not including depth statistics*

Description

This function extracts the PCA scores from the PCA results and plots the loadings. This function is to be use on output from the `pca_data()` function not including depth statistics.

Usage

```
pca_scores(
  pc_results = results,
  plot_loadings = TRUE,
  every_nth = 12,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

`pc_results`	PCA class object containing the output from the 'pca_results()' function.
`plot_loadings`	TRUE or FALSE. Plot PC loadings figures. Default TRUE.

every_nth	Numeric. Sequence of labels to show on mean power plot. Default is 12.
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Value

A data frame of pc scores containing one column for each Principal Component kept. If processing just one tag, the attribute 'unique_tag_ID' is given to the data frame with the tag_ID. Plots the PC loadings for each row of pc_data

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load pc_results
pc_results <- readRDS(file.path(filepath, "data/4_PCA/pc_results.rds"))

# Run pca_scores function
pc_scores <- pca_scores(
  pc_results = pc_results,
  plot_loadings = FALSE,
  every_nth = 12,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

---

plot_clusters *Plot the time-series depth records of the days closest to the centre of each cluster*

Description

plot_clusters plots the time-depth records of the days closest to the centre of each of the clusters. Each cluster is plotted both individually, and faceted together, with both a fixed y-axis and a free y-axis (depth).

Usage

```
plot_clusters(
  tag_vector = tag_list,
  data_folder = NULL,
  kmeans_result,
  No_days = 1,
```

```
  every_nth = 10,
  every_s = 0,
  Y_lim = c(0, 250, 50),
  color = TRUE,
  diel_shade = FALSE,
  dpi = 300,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

tag_vector	A character vector of tag IDs. E.g. 'c("123456", "456283", "AB98XJ").
data_folder	Parent folder path with separate folders for each tag data. E.g. "C:/Tag data". Defaults to 'data_dir'
kmeans_result	An object of class 'kmeans' containing the k-means clustering data. Output of 'k_clustering()' function.
No_days	Numerical. Number of days of each cluster to plot. Defaults to 1.
every_nth	Numerical. Optional down-sampling of data points to plot. Defaults to 10, plotting every 10th record.
every_s	Numerical. Alternative to every_nth. Optional down-sampling of data points to plot by number of seconds, as opposed to records. E.g. plots every 60th second, rather than 10th row of data. Must be a multiple of the sampling frequency. Overrides every_nth if != 0.
Y_lim	Character vector with minimum depth, maximum depth, and sequence for ticks on Y-axis. Must be numeric, positive and 3 elements long. E.g. c(0,1500,100).
color	TRUE or FALSE. Output clusters coloured by cluster assignment. Defaults to TRUE.
diel_shade	TRUE or FALSE. Output plot with night-time shading. Can be slow! Defaults to FALSE.
dpi	Numerical. DPI to use for 'ggsave()' output. E.g, 600
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Value

A plot list of all plots created of each cluster in the data. When output == TRUE this prints to file one figure for each Cluster with a fixed y-axis. Additionally outputs a facet plot of all clusters, and a free y-axis version of all plots.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load kmeans_result
kmeans_result <- readRDS(file.path(filepath, "data/5_k-means/kmeans_result.rds"))

# Run plot_clusters function
plot_clusters(
  tag_vector = "data",
  data_folder = filepath,
  kmeans_result = kmeans_result,
  No_days = 1,
  every_nth = 10,
  every_s = 0,
  Y_lim = c(0, 300, 50),
  color = TRUE,
  diel_shade = FALSE,
  dpi = 100,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

---

plot_cluster_TDR *Plot the time-series depth records of the selected tag. Colour days by cluster*

Description

plot_cluster_TDR plots the time-series depth record of the selected archival tag. Each day of data is coloured by the assigned cluster, this helps to visualise changes in vertical movement behaviour over time.

Usage

```
plot_cluster_TDR(
  tag_ID,
  data_folder = NULL,
  kmeans_result,
  every_nth = 10,
  every_s = 0,
  X_lim = NULL,
  Y_lim = c(0, 250, 50),
  date_breaks = "14 day",
  legend = TRUE,
  plot_size = c(12, 6),
```

```
  dpi = 300,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE,
  minmax_bins = 0
)
```

Arguments

tag_ID	Unique tag identification number in a vector of characters. E.g. "123456".
data_folder	Parent folder path with separate folders for each tag data. E.g. "C:/Tag data". Defaults to 'data_dir'
kmeans_result	An object of class 'kmeans' containing the k-means clustering data. Output of 'k_clustering()' function.
every_nth	Numerical. Optional down-sampling of data points to plot. Defaults to 10, plotting every 10th record.
every_s	Numerical. Alternative to every_nth. Optional down-sampling of data points to plot by number of seconds, as opposed to records. E.g. plots every 60th second, rather than 10th row of data. Must be a multiple of the sampling frequency. Overrides every_nth if != 0.
X_lim	Optional. Vector with two dates delimiting the time-depth record to plot. E.g. c("2000-01-01", "2000-11-23")
Y_lim	Character vector with minimum depth, maximum depth, and sequence for ticks on Y-axis. Must be numeric, positive and 3 elements long. E.g. c(0,1500,100).
date_breaks	X-axis ggplot2 date breaks. E.g, "24 hour, "3 day", "2 week".
legend	TRUE or FALSE. Whether or not to plot the figure legend. Defaults to TRUE.
plot_size	ggSave height and width for saving the output plot. Must be numeric, positive and 2 elements long. Default to 'c(12,6)'
dpi	Numerical. DPI to use for 'ggsave()' output. E.g, 600
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.
minmax_bins	Numerical. Alternative to every_nth and every_s for fast plotting of large archives. Splits the record into this many equal time bins and plots only the shallowest and deepest record in each, so dive extremes are kept. E.g. 2000. Overrides every_nth and every_s if != 0. Defaults to 0.

Value

Returns the cluster TDR plot. Additionally prints to file the TDR plot. Additionally outputs a facet plot of all tag_IDs.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load kmeans_result
kmeans_result <- readRDS(file.path(filepath, "data/5_k-means/kmeans_result.rds"))

# Run plot_clusters function
plot_cluster_TDR(
  tag_ID = "data",
  data_folder = filepath,
  kmeans_result = kmeans_result,
  every_nth = 10,
  every_s = 0,
  X_lim = NULL,
  Y_lim = c(0, 300, 50),
  date_breaks = "1 day",
  legend = TRUE,
  plot_size = c(12, 6),
  dpi = 100,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

---

plot_TDR *Plot the time-series depth dataset*

Description

This function plots the time-series depth data from the imported tag.

Usage

```
plot_TDR(
  rds_file,
  data_folder = NULL,
  every_nth = 20,
  every_s = 0,
  plot_size = c(12, 6),
  X_lim = NULL,
  Y_lim = c(0, 1500, 100),
  date_breaks = "14 day",
  dpi = 300,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE,
```

```
  minmax_bins = 0,
  colour_temp = FALSE
)
```

Arguments

rds_file	Character vector file path of rds file. E.g. ("E:/data/archive_days.rds")
data_folder	Parent folder path with separate folders for each tag data. E.g. "C:/Tag data". Defaults to 'data_dir'
every_nth	Numerical. Optional down-sampling of data points to plot. Defaults to 10, plotting every 10th record.
every_s	Numerical. Alternative to every_nth. Optional down-sampling of data points to plot by number of seconds, as opposed to records. E.g. plots every 60th second, rather than 10th row of data. Must be a multiple of the sampling frequency. Overrides every_nth if != 0.
plot_size	ggSave height and width for saving the output plot. Must be numeric, positive and 2 elements long. Default to 'c(12,6)'
X_lim	Optional. Vector with two dates delimiting the time-depth record to plot. E.g. c("2000-01-01", "2000-11-23")
Y_lim	Character vector with minimum depth, maximum depth, and sequence for ticks on Y-axis. Must be numeric, positive and 3 elements long. E.g. c(0,1500,100).
date_breaks	X-axis ggplot2 date breaks. E.g, "24 hour, "3 day", "2 week".
dpi	Numerical. DPI to use for 'ggsave()' output. E.g, 600
output	Logical. If TRUE, a plot file is saved to `output_folder`. Defaults to FALSE.
output_folder	Output folder path used when `output = TRUE`. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.
minmax_bins	Numerical. Alternative to every_nth and every_s for fast plotting of large archives. Splits the record into this many equal time bins and plots only the shallowest and deepest record in each, so dive extremes are kept. E.g. 2000. Overrides every_nth and every_s if != 0. Defaults to 0.
colour_temp	Logical. If TRUE, the depth record is coloured by temperature, from blue (cold) to red (warm). Needs a 'temp' column, i.e. the tag imported with `temp_col` in `import_tag_data()`. Defaults to FALSE.

Value

A data frame of plot data. The ggplot object is attached as the attribute 'TDR_plot'.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Run plot_TDR function
TDR_plot <- plot_TDR(
  rds_file = "data/archive_days.rds",
```

```
  data_folder = filepath,
  every_nth = 10,
  every_s = 0,
  plot_size = c(12, 6),
  X_lim = NULL,
  Y_lim = c(0, 300, 50),
  date_breaks = "24 hour",
  dpi = 100,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

`plot_time_at` *Plot time at depth or time at temperature*

Description

`plot_time_at` shows the percentage of time a tagged animal spent in each depth or temperature range, for the whole record, each month, or each day.

Usage

```
plot_time_at(
  archive,
  variable = "depth",
  breaks = NULL,
  by = "all",
  plot_size = c(8, 6),
  dpi = 300,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

`archive`	Data frame of time series depth data from `import_tag_data()`. Time at temperature needs a 'temp' column, i.e. the tag imported with `temp_col`.
`variable`	Character. "depth" for time at depth, or "temp" for time at temperature. Defaults to "depth".
`breaks`	Numerical. Limits of the depth (m) or temperature (degrees C) ranges, e.g. `seq(0, 300, by = 25)`. Defaults to NULL, which chooses about 12 ranges of round values covering the record.
`by`	Character. "all" for one histogram of the whole record, "month" for one histogram per month, or "day" for a heat map of each day. Defaults to "all".

plot_size	ggsave width and height for saving the plot. Must be numeric, positive and 2 elements long. Defaults to c(8, 6).
dpi	Numerical. DPI to use for 'ggsave()' output. Defaults to 300.
output	Logical. If TRUE, the plot is saved to output_folder, named e.g. 'time_at_depth.png' or 'time_at_temp_by_month.png'. Defaults to FALSE.
output_folder	Output folder path used when output = TRUE. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Details

Time is counted as the number of records in each range, so the record should have a constant sampling interval (see sampling_report()). Percentages are of the records in each group (the whole record, a month or a day) that have a value. Ranges include their lower limit, and the last range also includes its upper limit.

Records outside the breaks are left out, with a warning.

Value

A data frame with one row per group and range: 'group' ("All", the month, or the date), 'range' (e.g. "0 to 25"), its 'lower' and 'upper' limits, the number of 'records', the 'hours' they cover, and the 'percent' of the group's time. The ggplot object is attached as the attribute 'time_at_plot'.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load archive_days
archive_days <- readRDS(file.path(filepath, "data/archive_days.rds"))

# Time at depth over the whole record
tad <- plot_time_at(archive_days, variable = "depth", breaks = seq(0, 300, by = 25))
attr(tad, "time_at_plot")
head(tad)

# Time at depth each day
tad_daily <- plot_time_at(archive_days, by = "day")
attr(tad_daily, "time_at_plot")
```

---

run_app *Launch the FishDiveR app*

Description

run_app opens a point-and-click interface to FishDiveR in your web browser. The app runs locally on your computer, so your tag data never leave your machine.

Usage

```
run_app(project_folder = NULL, launch.browser = TRUE, ...)
```

Arguments

project_folder	Optional. Path of a project folder to open on launch. Defaults to NULL, choosing the folder in the app.
launch.browser	Logical. If TRUE, the app opens in your web browser. Defaults to TRUE.
...	Further arguments passed to shiny::runApp(), e.g. port.

Details

The app works on a project folder. Tags are listed in a tag table, saved as tag_metadata.csv in the project folder, and every step saves its results there: one sub-folder per tag, plus Combined_tags for the multi-tag PCA and k-means clustering. This is the same folder structure the FishDiveR functions use when output_folder is the project folder, so you can move between the app and R code at any point.

The app needs the 'shiny', 'bslib' and 'DT' packages. Install them with install.packages(c("shiny", "bslib", "DT")).

Value

Called for its side effect of running the app. Returns invisibly when the app is closed.

Examples

```
if (interactive()) {
  run_app()
}
```

---

sampling_report *Report irregular sampling in a depth record*

Description

sampling_report checks that a depth record has a constant sampling interval, and describes any irregularities: duplicate times, gaps, changes of sampling rate, intervals shorter than expected, times out of order, and days with fewer records than expected.

Usage

```
sampling_report(archive, interval_s = NULL, min_run = 10)

## S3 method for class 'sampling_report'
print(x, max_rows = 5, ...)
```

Arguments

archive	Data frame of time series depth data with a POSIXct 'date' column, e.g. from import_tag_data(). If it has a 'depth' column, records with no depth are left out.
interval_s	Numerical. Expected sampling interval in seconds. Defaults to the most common interval.
min_run	Numerical. Minimum number of consecutive intervals of the same length to count as a change of sampling rate. Defaults to 10.
x	A 'sampling_report' object.
max_rows	Numerical. Maximum number of each kind of irregularity, and of incomplete days, to list when printing. Defaults to 5.
...	Not used.

Details

The analysis steps of FishDiveR assume a constant sampling interval. Use this function on the output of import_tag_data(), which warns when the interval is not constant, to see where and what the irregularities are.

Each interval between consecutive records is compared with the expected interval, by default the most common one:

- **Duplicates**: two records with the same time.
- **Out of order**: a record timed before the previous record.
- **Rate changes**: at least min_run consecutive intervals of the same, different length, e.g. a tag switching from 5 to 75 second sampling.
- **Gaps**: other intervals longer than expected, i.e. missing records. on_grid is FALSE if the gap is not a whole number of expected intervals.
- **Short intervals**: other intervals shorter than expected, e.g. from timing errors.

Days are counted in the record's time zone, from the 'date_only' column if present. A day's percent compares its number of records with the number expected at the expected interval.

Value

A list of class 'sampling_report', printed as a summary, with:

interval_s The expected sampling interval in seconds.

regular TRUE if every interval equals interval_s.

n_records Number of records checked.

n_no_depth Number of records left out for having no depth.

time_zone Time zone used for days.

intervals Each interval length, with its number and percentage of intervals.

duplicates, out_of_order, rate_changes, gaps, short One row per irregularity, as described in Details.

days Records, expected records and percent complete for each day.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load archive_days, sampled every 60 seconds
archive_days <- readRDS(file.path(filepath, "data/archive_days.rds"))

# A regular record
sampling_report(archive_days)

# Remove 5 records to make a gap
report <- sampling_report(archive_days[-(5000:5004), ])
report
report$gaps
```

---

select_k *Perform k selection*

Description

select_k creates the elbow plot and silhouette width plot for assistance with selection of k

Usage

```
select_k(
  kmeans_data,
  standardise = TRUE,
  Max.k = 15,
  v_line = NULL,
  calc_gap = FALSE,
  plot_gap = FALSE,
  output = FALSE,
  output_folder = NULL,
  verbose = FALSE
)
```

Arguments

kmeans_data	Data frame containing the combined PC scores and depth statistics to perform k-means on. Output from the 'combine_data()' function.
standardise	TRUE or FALSE. Whether or not to standardise the data. Defaults to TRUE.
Max.k	Numerical. Maximum value of k to try. Defaults to 15.
v_line	Numerical. Option to add a vertical line to plot at a specific value of k. Defaults to NULL.
calc_gap	TRUE or FALSE. Whether or not to calculate the gap statistic. Defaults to FALSE

plot_gap	TRUE or FALSE. Whether or not to plot the gap statistic. Defaults to FALSE.
output	Logical. If TRUE, output is saved to output_folder. Defaults to FALSE.
output_folder	Output folder path. If output = TRUE, output_folder must be provided. Defaults to NULL.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Details

This function relies on random initialisation in k-means clustering. For reproducible results, users may wish to set a random seed prior to calling this function using set.seed().

Value

A 'ggplot' class object and creates a figure containing both the within-cluster sum of squares plot (elbow) and the average silhouette width plot for 1 to 'Max.k' clusters.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load kmeans_data
kmeans_data <- readRDS(file.path(filepath, "data/5_k-means/combined_stats.rds"))

# Run select_k function
selecting_k <- select_k(
  kmeans_data = kmeans_data,
  standardise = TRUE,
  Max.k = 8,
  v_line = 4,
  calc_gap = FALSE,
  plot_gap = FALSE,
  output = TRUE,
  output_folder = tempdir(),
  verbose = TRUE
)
```

---

thin_archive *Thin a depth record to a longer sampling interval*

Description

thin_archive averages a processed depth record into blocks of interval_s seconds, e.g. from 1 second to 15 second sampling. Use it on the archive passed to create_wavelet() to reduce the memory and time the wavelet analysis needs for long, high-frequency records.

Usage

```
thin_archive(archive, interval_s, verbose = FALSE)
```

Arguments

archive	Data frame of processed time series depth data from import_tag_data().
interval_s	Numerical. New sampling interval in seconds. Must be a multiple of the current sampling interval and divide a day evenly, e.g. 5, 10, 15, 30 or 60.
verbose	Logical. If TRUE, progress messages are shown. Defaults to FALSE.

Details

Each block of interval_s seconds becomes one record, timed at the start of the block, with the mean depth (and mean temperature, if present) of the block. Averaging, rather than keeping every nth record, stops movements faster than the new interval appearing as false longer cycles (aliasing).

Thinning only removes cycles shorter than about twice the new interval. As a rule of thumb, keep at least 20 records per cycle of the wavelet's lower period, i.e. interval_s <= lower_period_mins * 60 / 20: up to 15 seconds for a 5 minute lower period.

Keep using the full record for create_depth_stats(), as averaging slightly reduces maximum depths. Tags clustered together should all be thinned to the same interval.

Value

The thinned archive, with the same columns and 'time_zone' attribute. Returned unchanged if interval_s equals the current sampling interval.

Examples

```
# Set file path
filepath <- system.file("extdata", package = "FishDiveR")

# Load archive_days, sampled every 60 seconds
archive_days <- readRDS(file.path(filepath, "data/archive_days.rds"))

# Thin to 5 minute sampling
archive_5min <- thin_archive(archive_days, interval_s = 300, verbose = TRUE)
```

Index

combine_data, 2
create_depth_stats, 3
create_depth_stats(), *28*
create_wavelet, 5
create_wavelet(), *27*
create_wavelet_stats, 7

import_tag_data, 9
import_tag_data(), *21*, *22*, *25*, *28*

k_clustering, 10

pca_data, 12
pca_results, 13
pca_scores, 15
plot_cluster_TDR, 18
plot_clusters, 16
plot_TDR, 20
plot_time_at, 22
print.sampling_report
        *(*sampling_report*)*, 24

run_app, 23

sampling_report, 24
sampling_report(), *10*, *23*
select_k, 26

thin_archive, 27

WaveletComp::analyze.wavelet(), *6*
WaveletComp::wt.avg(), *6*