Package ‘animaltracker’

December 3, 2019

Title Animal Tracker

Version 0.1.0

Description Import, visualize, and analyze GPS and accelerometer data for spatial-temporal tracking of animals (e.g., cows).

Depends R (>= 3.5.0)

Imports zoo (>= 1.8.6), forcats (>= 0.4.0), lubridate (>= 1.7.0), tibble (>= 2.1.0), shinyBS (>= 0.61), V8 (>= 2.0), shinyjs (>= 1.0), shiny (>= 1.2.0), shinyWidgets (>= 0.4.4), shinycssloaders (>= 0.2.0), shinythemes (>= 1.1.2), leaflet (>= 2.0.2), leaflet.extras (>= 1.0.0), dplyr (>= 0.7.5), ggplot2 (>= 3.1.0), scales (>= 1.0.0), tidyr (>= 0.8.2), sp (>= 1.3.1), rgdal (>= 1.3.6), raster(>= 2.7.15), elevatr (>= 0.2.0), geosphere (>= 1.5.7)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.1

R topics documented:

`app_server` *Defines logic for updating the app based on user interaction in the ui*

Description

Defines logic for updating the app based on user interaction in the ui

Usage

```
app_server(input, output, session)
```

Arguments

`input`	see shiny app architecture
`output`	see shiny app architecture
`session`	see shiny app architecture

Value

server function for use in a shiny app

`app_ui` *Defines a user interface for the shiny app*

Description

Defines a user interface for the shiny app

Usage

```
app_ui()
```

Value

ui function for use in a shiny app

`boxplot_altitude` *Generates a boxplot to visualize the distribution of altitude by GPS.*

Description

Generates a boxplot to visualize the distribution of altitude by GPS.

Usage

```
boxplot_altitude(rds_path)
```

Arguments

`rds_path`	Path of .rds animal data file to read in

Value

overall boxplot of altitude by GPS

Examples

```
# Boxplot of altitude for demo data .rds

boxplot_altitude(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

boxplot_time_unit *Generates a boxplot to visualize the distribution of time between GPS measurements by GPS unit.*

Description

Generates a boxplot to visualize the distribution of time between GPS measurements by GPS unit.

Usage

```
boxplot_time_unit(rds_path)
```

Arguments

rds_path	Path of .rds animal data file to read in

Value

distribution of time between GPS measurements by GPS unit, as a boxplot

Examples

```
# Boxplot of GPS measurement time differences for demo data .rds

boxplot_time_unit(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

calc_bearing *Helper function for cleaning Columbus P-1 datasets. Given lat and long coords in degree decimal, convert to radians and compute bearing.*

Description

Helper function for cleaning Columbus P-1 datasets. Given lat and long coords in degree decimal, convert to radians and compute bearing.

Usage

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

Arguments

lat1	latitude of starting point
lon1	longitude of starting point
lat2	latitude of ending point
lon2	longitude of ending point

Value

bearing computed from given coordinates

clean_batch_df *Cleans a directory of animal data files*

Description

Cleans a directory of animal data files

Usage

```
clean_batch_df(data_info, filters = TRUE, tz_in = "UTC", tz_out = "UTC")
```

Arguments

`data_info`	list of animal data frames with information about the data, generated by store_batch
`filters`	filter bad data points, defaults to true
`tz_in`	input time zone, defaults to UTC
`tz_out`	output time zone, defaults to UTC

Value

clean df with all animal data files from the directory

clean_export_files *Cleans all animal GPS datasets (in .csv format) in a chosen directory. Optionally exports the clean data as spreadsheets, a single .rds data file, or as a list of data frames*

Description

Cleans all animal GPS datasets (in .csv format) in a chosen directory. Optionally exports the clean data as spreadsheets, a single .rds data file, or as a list of data frames

Usage

```
clean_export_files(
  data_dir,
  cleaned_filename = "animal_data.rds",
  cleaned_dir = "processed",
  tz_in = "UTC",
  tz_out = "UTC"
)
```

Arguments

`data_dir`	directory of GPS tracking files (in csv)
`cleaned_filename`	full name of output file (ending in .rds), defaults to data/animal_data.rds
`cleaned_dir`	directory to save the processed GPS datasets as spreadsheets (.csv), defaults to data/processed
`tz_in`	input time zone, defaults to UTC
`tz_out`	output time zone, defaults to UTC

Examples

```
# Clean all animal GPS .csv datasets in the demo directory

## Not run:
clean_export_files(system.file("extdata", "demo_nov19", package = "animaltracker"),
cleaned_filename = "ex_animal_data.rds", cleaned_dir = "clean_export_ex", tz = "UTC")

## End(Not run)
```

`clean_location_data` *Cleans a raw animal GPS dataset, implementing a standardized procedure to remove impossible values*

Description

Cleans a raw animal GPS dataset, implementing a standardized procedure to remove impossible values

Usage

```
clean_location_data(
  df,
  dtype,
  filters = TRUE,
  aniid = NA,
  gpsid = NA,
  maxrate = 84,
  maxcourse = 100,
  maxdist = 840,
  maxtime = 100,
  tz_in = "UTC",
  tz_out = "UTC"
)
```

Arguments

`df`	data frame in standardized format (e.g., from a raw spreadsheet)
`dtype`	data type, iGotU or Columbus P-1
`filters`	filter bad data points, defaults to true
`aniid`	identification code for the animal
`gpsid`	identification code for the GPS device
`maxrate`	maximum rate of travel (meters/minute) between consecutive points
`maxcourse`	maximum distance (meters) between consecutive points
`maxdist`	maximum geographic distance (meters) between consecutive points
`maxtime`	maximum time (minutes) between consecutive points
`tz_in`	input time zone, defaults to UTC
`tz_out`	output time zone, defaults to UTC

Examples

```
# Clean a data frame from csv

## Read igotU data
bannock_df <- read.csv(system.file("extdata", "demo_nov19/Bannock_2017_101_1149.csv",
package = "animaltracker"), skipNul=TRUE)

## Clean and filter
clean_location_data(bannock_df, dtype = "igotu", filters = TRUE, aniid = 1149,
gpsid = 101, maxrate = 84, maxdist = 840, maxtime = 100)

## Clean without filtering
clean_location_data(bannock_df, dtype = "igotu", filters = FALSE, aniid = 1149,
gpsid = 101, maxrate = 84, maxdist = 840, maxtime = 100)

# Clean a data frame from txt

## Read Columbus P-1 data
columbus_df <- read_columbus(system.file("extdata", "demo_columbus.TXT",
package = "animaltracker"))

## Clean and filter
clean_location_data(columbus_df, dtype = "columbus", filters = TRUE, aniid = 1149,
gpsid = 101, maxrate = 84, maxdist = 840, maxtime = 100)
```

`clean_store_batch` *Cleans a directory of animal data files and stores them locally in rds format*

Description

Cleans a directory of animal data files and stores them locally in rds format

Usage

```
clean_store_batch(
  data_info,
  filters = TRUE,
  zoom = 11,
  get_slope = TRUE,
  get_aspect = TRUE,
  min_lat = data_info$min_lat,
  max_lat = data_info$max_lat,
  min_long = data_info$min_long,
  max_long = data_info$max_long,
  tz_in = "UTC",
  tz_out = "UTC"
)
```

Arguments

`data_info`	list of animal data frames with information about the data, generated by store_batch

`filters`	filter bad data points, defaults to true
`zoom`	level of zoom, defaults to 11
`get_slope`	logical, whether to compute slope (in degrees), defaults to true
`get_aspect`	logical, whether to compute aspect (in degrees), defaults to true
`min_lat`	minimum latitude for filtering, defaults to min in data_info
`max_lat`	maximum latitude for filtering, defaults to max in data_info
`min_long`	minimum longitude for filtering, defaults to min in data_info
`max_long`	maximum longitude for filtering, defaults to max in data_info
`tz_in`	input time zone, defaults to UTC
`tz_out`	output time zone, defaults to UTC

Value

df of metadata for animal file directory

`compare_flags` *Joins and reformats two animal data frames for the purpose of flag comparison*

Description

Joins and reformats two animal data frames for the purpose of flag comparison

Usage

```
compare_flags(correct, candidate)
```

Arguments

`correct`	reference df
`candidate`	df to be compared to the reference

Value

joined and reformatted df

Examples

```
## Not run:
# Join and reformat unfiltered demo data and filtered demo data

## Get elevation data for unfiltered demo
unfiltered_elev <- lookup_elevation_aws(demo_unfiltered, zoom=1,
get_slope=FALSE, get_aspect=FALSE)

## Get elevation data for filtered demo
filtered_elev <- lookup_elevation_aws(demo_filtered, zoom=1, get_slope=FALSE, get_aspect=FALSE)

compare_flags(unfiltered_elev, filtered_elev)

## End(Not run)
```

compare_summarise_daily

Compares two animal datasets and calculates daily summary statistics by GPS GPS, date, lat, long, course, distance, rate, elevation column names should match.

Description

Compares two animal datasets and calculates daily summary statistics by GPS GPS, date, lat, long, course, distance, rate, elevation column names should match.

Usage

```
compare_summarise_daily(correct, candidate, out)
```

Arguments

`correct`	reference df
`candidate`	df to be compared to the reference
`out`	desired file name of .csv output summary

Value

summary df

Examples

```
# Compare and summarise unfiltered demo cows to filtered, grouped by both Date and GPS

## Not run:

## Get elevation data for unfiltered demo
unfiltered_elev <- lookup_elevation_aws(demo_unfiltered, zoom=1,
get_slope=FALSE, get_aspect=FALSE)

## Get elevation data for filtered demo
filtered_elev <- lookup_elevation_aws(demo_filtered, zoom=1, get_slope=FALSE, get_aspect=FALSE)

## Compare and summarise
compare_summarise_daily(unfiltered_elev, filtered_elev, "ex_compare_daily.csv")

## End(Not run)
```

compare_summarise_data

Compares two animal data frames and calculates summary statistics. GPS, date, lat, long, course, distance, rate, elevation column names should match.

Description

Compares two animal data frames and calculates summary statistics. GPS, date, lat, long, course, distance, rate, elevation column names should match.

Usage

```
compare_summarise_data(correct, candidate, gps_out, date_out)
```

Arguments

`correct`	reference df
`candidate`	df to be compared to the reference
`gps_out`	desired file name of .csv output summary by GPS collar
`date_out`	desired file name of .csv output summary by date

Value

list containing gps_out and date_out as dfs

Examples

```
# Compare and summarise unfiltered demo cows to filtered

## Not run:
## Get elevation data for unfiltered demo
unfiltered_elev <- lookup_elevation_aws(demo_unfiltered, zoom=1,
get_slope=FALSE, get_aspect=FALSE)

## Get elevation data for filtered demo
filtered_elev <- lookup_elevation_aws(demo_filtered, zoom=1, get_slope=FALSE, get_aspect=FALSE)

## Compare and summarise
compare_summarise_data(unfiltered_elev, filtered_elev, "ex_gps_compare.csv", "ex_date_compare.csv")

## End(Not run)
```

`deg_to_dec` *Helper function for cleaning Columbus P-1 datasets. Given lat or long coords in degrees and a direction, convert to decimal.*

Description

Helper function for cleaning Columbus P-1 datasets. Given lat or long coords in degrees and a direction, convert to decimal.

Usage

```
deg_to_dec(x, direction)
```

Arguments

`x`	lat or long coords in degrees
`direction`	direction of lat/long

Value

converted x

`demo` *Demo animal GPS data from cows*

Description

Demo animal GPS data from cows

Usage

```
demo
```

Format

A data frame with 2171 rows and 29 variables

`demo_comparison` *Demo comparison of two animal datasets*

Description

Demo comparison of two animal datasets

Usage

```
demo_comparison
```

Format

A data frame with 2758 rows and 33 variables

`demo_filtered` *Filtered demo animal GPS data from cows*

Description

Filtered demo animal GPS data from cows

Usage

```
demo_filtered
```

Format

A data frame with 2187 rows and 26 variables

`demo_info` *Raw demo animal GPS data from cows with information*

Description

Raw demo animal GPS data from cows with information

Usage

```
demo_info
```

Format

A list with 10 elements

`demo_meta` *Metadata for demo animal GPS data from cows*

Description

Metadata for demo animal GPS data from cows

Usage

```
demo_meta
```

Format

A data frame with 6 rows and 11 variables

demo_unfiltered *Unfiltered demo animal GPS data from cows*

Description

Unfiltered demo animal GPS data from cows

Usage

```
demo_unfiltered
```

Format

A data frame with 2288 rows and 32 variables

detect_peak_modz *Alternative implementation of the robust peak detection algorithm by van Brakel 2014 Classifies data points with modified z-scores greater than max_score as outliers ccording to Iglewicz and Hoaglin 1993*

Description

Alternative implementation of the robust peak detection algorithm by van Brakel 2014 Classifies data points with modified z-scores greater than max_score as outliers ccording to Iglewicz and Hoaglin 1993

Usage

```
detect_peak_modz(df_comparison, lag = 5, max_score = 3.5)
```

Arguments

`df_comparison`	output of compare_flags
`lag`	width of interval to compute rolling median and MAD, defaults to 5
`max_score`	modified z-score cutoff to classify observations as outliers, defaults to 3.5

Value

df with classifications

Examples

```
## Not run:
# Join and reformat unfiltered demo data and filtered demo data

## Get elevation data for unfiltered demo
unfiltered_elev <- lookup_elevation_aws(demo_unfiltered, zoom=1,
get_slope=FALSE, get_aspect=FALSE)
```

```
## Get elevation data for filtered demo
filtered_elev <- lookup_elevation_aws(demo_filtered, zoom=1, get_slope=FALSE, get_aspect=FALSE)

## Get comparison df
comparison <- compare_flags(unfiltered_elev, filtered_elev)

detect_peak_modz(comparison, lag = 5, max_score = 3.5)

## End(Not run)
```

dev_add_to_gitignore *Add big files to a .gitignore file*

Description

Add big files to a .gitignore file

Usage

```
dev_add_to_gitignore(data_dir)
```

Arguments

data_dir	directory of animal data files

Examples

```
# Detect large files in the demo directory and add to the .gitignore file
## Not run:
dev_add_to_gitignore(system.file("extdata", "demo_nov19", package = "animaltracker"))

## End(Not run)
```

get_data_from_meta *Get animal data set from specified meta. If date range is invalid, automatically returns all animal data specified by meta_df.*

Description

Get animal data set from specified meta. If date range is invalid, automatically returns all animal data specified by meta_df.

Usage

```
get_data_from_meta(meta_df, min_date, max_date)
```

Arguments

meta_df	data frame of specified meta
min_date	minimum date specified by user
max_date	maximum date specified by user

get_file_meta *Generate metadata for a directory of animal data files*

Description

Generate metadata for a directory of animal data files

Usage

```
get_file_meta(data_dir)
```

Arguments

data_dir	directory of animal data files

Value

list of data info as a list of animal IDs and GPS units

Examples

```
# Get metadata for demo directory

get_file_meta(system.file("extdata", "demo_nov19", package = "animaltracker"))
```

get_meta *Generate metadata for an animal data frame - filename, site, date min/max, animals, min/max lat/longitude, storage location*

Description

Generate metadata for an animal data frame - filename, site, date min/max, animals, min/max lat/longitude, storage location

Usage

```
get_meta(df, file_id, file_name, site, ani_id, storage_loc)
```

Arguments

df	clean animal data frame
file_id	ID number of .csv source of animal data frame
file_name	.csv source of animal data frame
site	physical source of animal data
ani_id	ID of animal found in data frame
storage_loc	.rds storage location of animal data frame

Value

df of metadata for animal data frame

`histogram_animal_elevation` *Generate a histogram of the distribution of modeled elevation - measured altitude*

Description

Generate a histogram of the distribution of modeled elevation - measured altitude

Usage

```
histogram_animal_elevation(datapts)
```

Arguments

`datapts`	GPS data with measured Altitude and computed Elevation data

Value

histogram of the distribution of modeled elevation - measured altitude

Examples

```
# Histogram of elevation - altitude for the demo data

histogram_animal_elevation(demo)
```

`histogram_time` *Generates a histogram to visualize the distribution of time between GPS measurements.*

Description

Generates a histogram to visualize the distribution of time between GPS measurements.

Usage

```
histogram_time(rds_path)
```

Arguments

`rds_path`	Path of .rds cow data file to read in

Value

distribution of time between GPS measurements, as a histogram

Examples

```
# Histogram of GPS measurement time differences for demo data .rds

histogram_time(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

`histogram_time_unit` *Generates a histogram to visualize the distribution of time between GPS measurements by GPS unit.*

Description

Generates a histogram to visualize the distribution of time between GPS measurements by GPS unit.

Usage

```
histogram_time_unit(rds_path)
```

Arguments

`rds_path` Path of .rds animal data file to read in

Value

distribution of time between GPS measurements by GPS unit, as a histogram

Examples

```
# Histogram of GPS measurement time differences by GPS unit for demo data .rds

histogram_time_unit(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

`join_summaries` *Joins two animal data frame summaries by a column and appends differences*

Description

Joins two animal data frame summaries by a column and appends differences

Usage

```
join_summaries(correct_summary, candidate_summary, by_str, daily = F)
```

Arguments

`correct_summary`
summary df of reference dataset, returned by summarise_anidf

`candidate_summary`
summary df of dataset to be compared to reference, returned by summarise_anidf

`by_str` column to join by as a string, null if daily=T

`daily` whether to group by both GPS and Date for daily summary, defaults to False

Examples

```
# Join date summaries of unfiltered and filtered demo data

## Not run:

## Get elevation data for unfiltered demo
unfiltered_elev <- lookup_elevation_aws(demo_unfiltered, zoom=1,
get_slope=FALSE, get_aspect=FALSE)

## Get elevation data for filtered demo
filtered_elev <- lookup_elevation_aws(demo_filtered, zoom=1, get_slope=FALSE, get_aspect=FALSE)

## Summarise unfiltered demo by date
unfiltered_summary <- summarise_anidf(unfiltered_elev, Date, Latitude, Longitude,
Distance, Course, Rate, Elevation, daily=FALSE)

## Summarise filtered demo by date
filtered_summary <- summarise_anidf(filtered_elev, Date, Latitude, Longitude,
Distance, Course, Rate, Elevation, daily=FALSE)

## Join
join_summaries(unfiltered_summary, filtered_summary, "Date", daily=F)


## End(Not run)
```

---

line_compare *Compares moving averages of a variable for two datasets over time, grouped by GPS GPS, Date, and col columns should match*

Description

Compares moving averages of a variable for two datasets over time, grouped by GPS GPS, Date, and col columns should match

Usage

```
line_compare(correct, candidate, col, out)
```

Arguments

correct	reference df
candidate	df to be compared to the reference
col	variable to plot the moving average for
out	file name to save plot

Value

faceted line plot of moving averages over time grouped by GPS

Examples

```
# Faceted line plot comparing moving averages over time
# grouped by GPS for unfiltered and filtered demo data

## Not run:
## Set distance as the y axis
line_compare(demo_unfiltered, demo_filtered, Distance, "ex_line_dist.png")

## End(Not run)
```

`lookup_elevation_aws` *Add elevation data from public AWS terrain tiles to long/lat coordinates of animal gps data*

Description

Add elevation data from public AWS terrain tiles to long/lat coordinates of animal gps data

Usage

```
lookup_elevation_aws(anidf, zoom = 11, get_slope = TRUE, get_aspect = TRUE)
```

Arguments

`anidf`	animal tracking dataframe
`zoom`	level of zoom, defaults to 11
`get_slope`	logical, whether to compute slope (in degrees), defaults to true
`get_aspect`	logical, whether to compute aspect (in degrees), defaults to true

Value

original data frame, with Elevation column appended

Examples

```
# Add elevation data to filtered demo data frame

## Not run:
## Lookup with slope and aspect
lookup_elevation_aws(demo_filtered, zoom = 11, get_slope = TRUE, get_aspect = TRUE)

## End(Not run)
```

lookup_elevation_file *Add elevation data from terrain tiles to long/lat coordinates of animal gps data*

Description

Add elevation data from terrain tiles to long/lat coordinates of animal gps data

Usage

```
lookup_elevation_file(
  elev,
  anidf,
  zoom = 11,
  get_slope = TRUE,
  get_aspect = TRUE
)
```

Arguments

elev	elevation data as raster
anidf	animal tracking dataframe
zoom	level of zoom, defaults to 11
get_slope	logical, whether to compute slope (in degrees), defaults to true
get_aspect	logical, whether to compute aspect (in degrees), defaults to true

Value

original data frame, with terrain column(s) appended

process_elevation *Export modeled elevation data from existing animal data file*

Description

Export modeled elevation data from existing animal data file

Usage

```
process_elevation(
  zoom = 11,
  get_slope = TRUE,
  get_aspect = TRUE,
  in_path,
  out_path
)
```

Arguments

`zoom`	level of zoom, defaults to 11
`get_slope`	logical, whether to compute slope (in degrees), defaults to true
`get_aspect`	logical, whether to compute aspect (in degrees), defaults to true
`in_path`	animal tracking data file to model elevation from
`out_path`	exported file path, .rds

Value

list of data frames with gps data augmented by elevation

Examples

```
# Export elevation data from demo .rds datasets

## Not run:
process_elevation(zoom = 11, get_slope = TRUE, get_aspect = TRUE,
in_path = system.file("extdata", "demo_nov19.rds",
package = "animaltracker"), out_path = "demo_nov19_elev.rds")


## End(Not run)
```

`qqplot_time` *Generates a QQ plot to show the distribution of time between GPS measurements.*

Description

Generates a QQ plot to show the distribution of time between GPS measurements.

Usage

```
qqplot_time(rds_path)
```

Arguments

`rds_path`	Path of .rds animal data file to read in

Value

quantile-quantile plot to show distribution of time between GPS measurements

Examples

```
# QQ plot of GPS measurment time differences for demo data .rds

qqplot_time(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

quantile_time *Determines the GPS measurement time value difference values roughly corresponding to quantiles with .05 intervals.*

Description

Determines the GPS measurement time value difference values roughly corresponding to quantiles with .05 intervals.

Usage

```
quantile_time(rds_path)
```

Arguments

rds_path	Path of .rds animal data file to read in

Value

approximate time difference values corresponding to quantiles (.05 intervals)

Examples

```
# Read in .rds of demo data and calculate time difference quantiles

quantile_time(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

read_columbus *Read and process a Columbus P-1 data file containing NMEA records into a data frame*

Description

Read and process a Columbus P-1 data file containing NMEA records into a data frame

Usage

```
read_columbus(filename)
```

Arguments

filename	path of Columbus P-1 data file

Value

NMEA records in RMC and GGA formats as a data frame

Examples

```
## Not run:
read_columbus(system.file("extdata", "demo_columbus.TXT", package = "animaltracker"))

## End(Not run)
```

`read_gps` *Reads a GPS dataset of unknown format at location filename*

Description

Reads a GPS dataset of unknown format at location filename

Usage

```
read_gps(filename)
```

Arguments

`filename`	location of the GPS dataset

Value

list containing the dataset as a df and the format

`read_zip_to_rasters` *Read an archive of altitude mask files and convert the first file into a raster object*

Description

Read an archive of altitude mask files and convert the first file into a raster object

Usage

```
read_zip_to_rasters(filename, exdir = "inst/extdata/elev")
```

Arguments

`filename`	path of altitude mask file archive
`exdir`	path to extract files

Value

the first altitude mask file as a raster object

run_shiny_animaltracker *You can run the animaltracker Shiny app by calling this function.*

Description

You can run the animaltracker Shiny app by calling this function.

Usage

```
run_shiny_animaltracker(browser = TRUE, showcase = FALSE)
```

Arguments

browser	logical, whether to launch the app in your default browser (defaults to TRUE)
showcase	logical, whether to launch the app in 'showcase' mode (defaults to FALSE)

Examples

```
## Not run:
# Run the animaltracker app
run_shiny_animaltracker()

## End(Not run)
```

run_validation_app *Run the Shiny validation app*

Description

Run the Shiny validation app

Usage

```
run_validation_app()
```

save_meta — *Save metadata to a data frame and return it*

Description

Save metadata to a data frame and return it

Usage

```
save_meta(meta_df, file_meta)
```

Arguments

meta_df	the data frame to store metadata in
file_meta	meta for a .csv file generated by get_meta

store_batch_list — *Generates basic metadata about a directory of animal data files and stores the files as data frames as a list with the meta*

Description

Generates basic metadata about a directory of animal data files and stores the files as data frames as a list with the meta

Usage

```
store_batch_list(data_dir)
```

Arguments

data_dir	location of animal data files, in list format

Value

a list of animal data frames with information about the data

summarise_anidf *Calculates summary statistics for an animal data frame*

Description

Calculates summary statistics for an animal data frame

Usage

```
summarise_anidf(anidf, by, lat, long, dist, course, rate, elev, daily = F)
```

Arguments

anidf	the animal data frame
by	column to group by, null if daily=T
lat	latitude column
long	longitude column
dist	distance column
course	course column
rate	rate column
elev	elevation column
daily	whether to group by both GPS and Date for daily summary, defaults to False

Examples

```
# Summary of demo data by date

summarise_anidf(demo, Date, Latitude, Longitude, Distance, Course, Rate, Elevation, daily=FALSE)
```

summarise_col *Get summary statistics for a single column in an animal data frame*

Description

Get summary statistics for a single column in an animal data frame

Usage

```
summarise_col(df, col)
```

Arguments

df	animal data frame
col	column to get summary stats for, as a string

Value

data frame of summary stats for col

Examples

```
# Get summary statistics for Distance column of demo data

summarise_col(demo, Distance)
```

---

summarise_unit *Summarise a number of animal datasets by GPS unit*

Description

Summarise a number of animal datasets by GPS unit

Usage

```
summarise_unit(rds_path)
```

Arguments

rds_path	Path of .rds cow data file to read in

Value

summary statistics for animals by GPS unit

Examples

```
# Read in .rds of demo data and summarise by GPS unit

summarise_unit(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

---

violin_compare *Compares summary statistics from two datasets as side-by-side violin plots*

Description

Compares summary statistics from two datasets as side-by-side violin plots

Usage

```
violin_compare(df_summary, by, col_name, out)
```

Arguments

`df_summary`	data frame of summary statistics from both datasets to be compared
`by`	GPS or Date
`col_name`	variable in df_summary to be used for the y-axis, as a string
`out`	file name to save plot

Value

side-by-side violin plots

Examples

```
# Violin plot comparing unfiltered and filtered demo data summaries by date for a single variable

## Not run:

## Get elevation data for unfiltered demo
unfiltered_elev <- lookup_elevation_aws(demo_unfiltered, zoom=1,
get_slope=FALSE, get_aspect=FALSE)

## Get elevation data for filtered demo
filtered_elev <- lookup_elevation_aws(demo_filtered, zoom=1, get_slope=FALSE, get_aspect=FALSE)

## Summarise unfiltered demo
unfiltered_summary <- summarise_anidf(unfiltered_elev, Date, Latitude, Longitude,
Distance, Course, Rate, Elevation, daily=FALSE)

## Summarise filtered demo
filtered_summary <- summarise_anidf(filtered_elev, Date, Latitude, Longitude,
Distance, Course, Rate, Elevation, daily=FALSE)

## Join
summary <- join_summaries(unfiltered_summary, filtered_summary, "Date", daily=FALSE)

## Violin plot

violin_compare(summary, Date, "meanElev", "ex_elev_violin.png")


## End(Not run)
```

Index