# Package ‘animaltracker’

March 25, 2020

**Title**  Animal Tracker  

**Version**  0.1.0  

**Description** Utilities for spatial-temporal analysis and visualization of animal (e.g. cattle) tracking data. The core feature is a 'shiny' web application for customized processing of GPS logs, including features for data augmentation (e.g. elevation lookup), data selection, export, plotting, and statistical summaries. A data validation application allows for side-by-side comparison via time series plots and extreme value detection described by J.P. van Brakel [https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data/].

**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**  7.0.2

**NeedsCompilation**  no

**Author**  Joe Champion [aut, cre], 
Thea Sukianto [aut], 
Chithkala Dhulipati [aut]

**Maintainer**  Joe Champion <joechampion@boisestate.edu>

**Repository**  CRAN

**Date/Publication**  2020-03-25 11:50:02 UTC
# R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_server</td>
<td>3</td>
</tr>
<tr>
<td>app_ui</td>
<td>3</td>
</tr>
<tr>
<td>boxplot_altitude</td>
<td>4</td>
</tr>
<tr>
<td>boxplot_time_unit</td>
<td>4</td>
</tr>
<tr>
<td>calc_bearing</td>
<td>5</td>
</tr>
<tr>
<td>clean_batch_df</td>
<td>5</td>
</tr>
<tr>
<td>clean_export_files</td>
<td>6</td>
</tr>
<tr>
<td>clean_location_data</td>
<td>7</td>
</tr>
<tr>
<td>clean_store_batch</td>
<td>8</td>
</tr>
<tr>
<td>compare_flags</td>
<td>9</td>
</tr>
<tr>
<td>compare_summarise_daily</td>
<td>10</td>
</tr>
<tr>
<td>compare_summarise_data</td>
<td>11</td>
</tr>
<tr>
<td>deg_to_dec</td>
<td>12</td>
</tr>
<tr>
<td>demo</td>
<td>12</td>
</tr>
<tr>
<td>demo_comparison</td>
<td>13</td>
</tr>
<tr>
<td>demo_filtered</td>
<td>13</td>
</tr>
<tr>
<td>demo_filtered_elev</td>
<td>13</td>
</tr>
<tr>
<td>demo_info</td>
<td>14</td>
</tr>
<tr>
<td>demo_meta</td>
<td>14</td>
</tr>
<tr>
<td>demo_unfiltered</td>
<td>14</td>
</tr>
<tr>
<td>demo_unfiltered_elev</td>
<td>15</td>
</tr>
<tr>
<td>detect_peak_modz</td>
<td>15</td>
</tr>
<tr>
<td>dev_add_to_gitignore</td>
<td>16</td>
</tr>
<tr>
<td>get_data_from_meta</td>
<td>16</td>
</tr>
<tr>
<td>get_file_meta</td>
<td>17</td>
</tr>
<tr>
<td>get_meta</td>
<td>17</td>
</tr>
<tr>
<td>histogram_animal_elevation</td>
<td>18</td>
</tr>
<tr>
<td>histogram_time</td>
<td>18</td>
</tr>
<tr>
<td>histogram_time_unit</td>
<td>19</td>
</tr>
<tr>
<td>join_summaries</td>
<td>20</td>
</tr>
<tr>
<td>line_compare</td>
<td>21</td>
</tr>
<tr>
<td>lookup_elevation_aws</td>
<td>21</td>
</tr>
<tr>
<td>lookup_elevation_file</td>
<td>22</td>
</tr>
<tr>
<td>process_elevation</td>
<td>23</td>
</tr>
<tr>
<td>qqplot_time</td>
<td>23</td>
</tr>
<tr>
<td>quantile_time</td>
<td>24</td>
</tr>
<tr>
<td>read_columbus</td>
<td>25</td>
</tr>
<tr>
<td>read_gps</td>
<td>25</td>
</tr>
<tr>
<td>read_zip_to_rasters</td>
<td>26</td>
</tr>
<tr>
<td>run_shiny_animaltracker</td>
<td>26</td>
</tr>
<tr>
<td>run_validation_app</td>
<td>27</td>
</tr>
<tr>
<td>save_meta</td>
<td>27</td>
</tr>
<tr>
<td>store_batch_list</td>
<td>28</td>
</tr>
<tr>
<td>summarise_anidf</td>
<td>28</td>
</tr>
<tr>
<td>summarise_col</td>
<td>29</td>
</tr>
<tr>
<td>summarise_unit</td>
<td>30</td>
</tr>
</tbody>
</table>
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
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**

```r
# Boxplot of altitude for demo data .rds
boxplot_altitude(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

---

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**

```r
# Boxplot of time unit for demo data .rds
boxplot_time_unit(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```
**calc_bearing**

**Examples**

```r
# Boxplot of GPS measurement time differences for demo data .rds
boxplot_time_unit(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

**Description**

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

**Usage**

```r
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**

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

Arguments

data_dir directory of GPS tracking files (in csv)
tz_in input time zone, defaults to UTC
tz_out output time zone, defaults to UTC
export logical, whether to export the clean data, defaults to False
cleaned_filename full name of output file (ending in .rds) when export is True
cleaned_dir directory to save the processed GPS datasets as spreadsheets (.csv) when export is True

Value

clean df with all animal data files from the directory

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,
tz_in = "UTC",  
tz_out = "UTC",  
export = FALSE,
cleaned_filename = NULL,
cleaned_dir = NULL  
)

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

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,
tz_in = "UTC",  
tz_out = "UTC",  
export = FALSE,
cleaned_filename = NULL,
cleaned_dir = NULL  
)

Arguments

data_dir directory of GPS tracking files (in csv)
tz_in input time zone, defaults to UTC
tz_out output time zone, defaults to UTC
export logical, whether to export the clean data, defaults to False
cleaned_filename full name of output file (ending in .rds) when export is True
cleaned_dir directory to save the processed GPS datasets as spreadsheets (.csv) when export is True

Value

clean df with all animal data files from the directory

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,
tz_in = "UTC",  
tz_out = "UTC",  
export = FALSE,
cleaned_filename = NULL,
cleaned_dir = NULL  
)
**clean_location_data**

**Examples**

```r
# Clean all animal GPS .csv datasets in the demo directory
clean_export_files(system.file("extdata", "demo_nov19", package = "animaltracker"))
```

**Description**

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

**Usage**

```r
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
Value

df of clean animal GPS data

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,
compare_flags

    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

Description

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

Usage

compare_flags(correct, candidate, elev = TRUE, slope = TRUE)

Arguments

- correct: reference df
- candidate: df to be compared to the reference
- elev: logical, whether to include elevation, defaults to true
- slope: logical, whether to include slope, defaults to true

Value

joined and reformatted df
Examples

# Join and reformat unfiltered demo data and filtered demo data
compare_flags(demo_unfiltered_elev, demo_filtered_elev)

compare_summarise_daily

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

Description

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

Usage

compare_summarise_daily(correct, candidate, export = TRUE, out = NULL)

Arguments

correct reference df
candidate df to be compared to the reference
export logical, whether to export summary to .csv, defaults to False
out desired file name of .csv output summary when export is True

Value

summary df

Examples

# Compare and summarise unfiltered demo cows to filtered, grouped by both Date and GPS
compare_summarise_daily(demo_unfiltered_elev, demo_filtered_elev)
**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

```r
compare_summarise_data(
  correct,
  candidate,
  export = FALSE,
  gps_out = NULL,
  date_out = NULL
)
```

### Arguments

- **correct**: reference df
- **candidate**: df to be compared to the reference
- **export**: logical, whether to export summaries to .csv, defaults to False
- **gps_out**: desired file name of .csv output summary by GPS collar when export is True
- **date_out**: desired file name of .csv output summary by date when export is True

### Value

list containing gps_out and date_out as dfs

### Examples

```r
# Compare and summarise unfiltered demo cows to filtered
compare_summarise_data(demo_unfiltered_elev, demo_filtered_elev)
```
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_filtered_elev
Filtered demo animal GPS data from cows with elevation appended at zoom 1

Description
Filtered demo animal GPS data from cows with elevation appended at zoom 1

Usage
demo_filtered_elev

Format
A data frame with 2187 rows and 29 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
demo_unfiltered_elev  

**Description**

Unfiltered demo animal GPS data from cows with elevation appended at zoom 1

**Usage**

demo_unfiltered_elev

**Format**

A data frame with 2288 rows and 35 variables

detect_peak_modz  

**Description**

Alternative implementation of the robust peak detection algorithm by van Brakel 2016

**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**

# Join and reformat unfiltered demo data and filtered demo data

detect_peak_modz(demo_comparison, lag = 5, max_score = 3.5)
get_data_from_meta

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

Value

df of animal data from specified meta

dev_add_to_gitignore

Description

Add big files to a .gitignore file

Usage

dev_add_to_gitignore(data_dir)

Arguments

- data_dir: directory of animal data files

Value

None
**get_file_meta**

*Generate metadata for a directory of animal data files*

**Description**

Generate metadata for a directory of animal data files

**Usage**

```r
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**

```r
# 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**

```r
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
**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**

```r
histogram_animal_elevation(datapts)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datapts</td>
<td>GPS data with measured Altitude and computed Elevation data</td>
</tr>
</tbody>
</table>

**Value**

histogram of the distribution of modeled elevation - measured altitude

**Examples**

```r
# Histogram of elevation - altitude for the demo data
demo
data = read_rds_demo()

histogram_animal_elevation(data)
```

---

**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**

```r
histogram_time(rds_path)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rds_path</td>
<td>Path of .rds cow data file to read in</td>
</tr>
</tbody>
</table>
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"))

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 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 = FALSE)

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=TRUE

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

Value

df of joined summaries with differences

Examples

# Join date summaries of unfiltered and filtered demo data
## Summarise unfiltered demo by date
unfiltered_summary <- summarise_anidf(demo_unfiltered_elev, Date, Latitude, Longitude, Distance, Course, Rate, Elevation, daily=FALSE)

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

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

Description

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

Usage

line_compare(correct, candidate, col, export = FALSE, out = NULL)

Arguments

correct reference df
candidate df to be compared to the reference
col variable to plot the moving average for
export logical, whether to export plot, defaults to False
out .png file name to save plot when export is True

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
## Set distance as the y axis
line_compare(demo_unfiltered, demo_filtered, Distance)

lookup_elevation_aws

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)
lookup_elevation_file

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

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**

Process and optionally export modeled elevation data from existing animal data file

**Description**

Process and optionally export modeled elevation data from existing animal data file.

**Usage**

```r
process_elevation(
  zoom = 11,
  get_slope = TRUE,
  get_aspect = TRUE,
  in_path,
  export = FALSE,
  out_path = NULL
)
```

**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
- `export`: logical, whether to export data with elevation, defaults to False
- `out_path`: .rds file path for processed data when export is True

**Value**

list of data frames with gps data augmented by elevation

---

**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**

```r
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 measurement 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

read_columbus(system.file("extdata", "demo_columbus.TXT", package = "animaltracker"))

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**
```r
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  
*Run the animaltracker 'shiny' app by calling this function. Depending on the size of input files, it may be advisable to increase the maximum request size.*

**Description**
Run the animaltracker 'shiny' app by calling this function. Depending on the size of input files, it may be advisable to increase the maximum request size.

**Usage**
```r
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)

**Value**
None
run_validation_app

Run the 'shiny' validation app. Depending on the size of input files, it may be advisable to increase the maximum request size.

Description

Run the 'shiny' validation app. Depending on the size of input files, it may be advisable to increase the maximum request size.

Usage

run_validation_app()

Value

None

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

Value

df of metadata
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 = FALSE)

Arguments
- anidf: the animal data frame
- by: column to group by, null if daily=TRUE
- 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
summarise_col

Value

df of summary statistics for the animal data frame

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**

```r
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**

```r
# 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**

```r
violin_compare(df_summary, by, col_name, export = FALSE, out = NULL)
```

**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
- `export`
  logical, whether to export plot, defaults to False
- `out`
  .png file name to save plot when export is True
Value

side-by-side violin plots

Examples

# Violin plot comparing unfiltered and filtered demo data summaries by date for a single variable  
## Summarise unfiltered demo  
unfiltered_summary <- summarise_anidf(demo_unfiltered_elev, Date, Latitude, Longitude, Distance, Course, Rate, Elevation, daily=FALSE)

## Summarise filtered demo  
filtered_summary <- summarise_anidf(demo_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")
Index

*Topic datasets
  demo, 12
  demo_comparison, 13
  demo_filtered, 13
  demo_filtered_elev, 13
  demo_info, 14
  demo_meta, 14
  demo_unfiltered, 14
  demo_unfiltered_elev, 15

app_server, 3
app_ui, 3

boxplot_altitude, 4
boxplot_time_unit, 4

calc_bearing, 5
clean_batch_df, 5
clean_export_files, 6
clean_location_data, 7
clean_store_batch, 8
compare_flags, 9
compare_summarise_daily, 10
compare_summarise_data, 11

deg_to_dec, 12
demo, 12
demo_comparison, 13
demo_filtered, 13
demo_filtered_elev, 13
demo_info, 14
demo_meta, 14
demo_unfiltered, 14
demo_unfiltered_elev, 15
detect_peak_modz, 15
dev_add_to_gitignore, 16

get_data_from_meta, 16
get_file_meta, 17
get_meta, 17

histogram_animal_elevation, 18
histogram_time, 18
histogram_time_unit, 19
join_summaries, 20
line_compare, 21
lookup_elevation_aws, 21
lookup_elevation_file, 22
process_elevation, 23
qqplot_time, 23
quantile_time, 24
read_columbus, 25
read_gps, 25
read_zip_to_rasters, 26
run_shiny_animaltracker, 26
run_validation_app, 27
save_meta, 27
store_batch_list, 28
summarise_anidf, 28
summarise_col, 29
summarise_unit, 30
violin_compare, 30