Package ‘animaltracker’

November 17, 2020

Title Animal Tracker

Version 0.2.0

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 httr (>= 1.4.0), maptools (>= 1.0.0), zoo (>= 1.8.6), forcats (>= 0.4.0), lubridate (>= 1.7.0), tibble (>= 2.1.0), shinyBS (>= 0.61), shinyjs (>= 2.0.0), shiny (>= 1.2.0), shinyWidgets (>= 0.4.4), shinyccslayers (>= 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), geosphere (>= 1.5.7)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Joe Champion [aut, cre], Thea Sukianto [aut]

Maintainer Joe Champion <joechampion@boisestate.edu>

Repository CRAN

Date/Publication 2020-11-17 17:20:06 UTC

R topics documented:

app_server ................................................................. 3
R topics documented:

app.ui ..................................................  4
boxplot_altitude ......................................  4
boxplot_time_unit ....................................  5
calc_bearing .........................................  5
clean_batch_df .......................................  6
clean_export_files ...................................  6
clean_location_data ..................................  7
clean_store_batch ....................................  9
compare_flags ........................................ 10
compare_summarise_daily ......................... 11
compare_summarise_data ......................... 12
datePicker ............................................. 13
datePickerOutput .................................... 13
deg_to_dec ........................................... 14
demo ..................................................... 14
demo_comparison .................................... 15
demo_filtered ........................................ 15
demo_filtered_elev ................................ 15
demo_info ............................................ 16
demo_meta ............................................ 16
demo_unfiltered .................................... 16
demo_unfiltered_elev ................................ 17
detect_peak_modz .................................... 17
dev_add_to_gitignore ............................... 18
get_data_from_meta ................................ 18
get_file_meta ........................................ 19
get_meta .............................................. 19
histogram_animal_elevation ................... 20
histogram_time ...................................... 20
histogram_time_unit ................................ 21
join_summaries ...................................... 22
line_compare ......................................... 23
lookup_elevation_aws ............................. 23
lookup_elevation_file ............................. 24
process_elevation .................................. 25
qqplot_time .......................................... 25
quantile_time ....................................... 26
reactivePicker ....................................... 27
reactivePickerOutput .............................. 28
reactivePlot .......................................... 28
reactivePlotOutput .................................. 29
reactiveRange ....................................... 29
reactiveRangeOutput ............................... 30
read_columbus ...................................... 30
read_gps ............................................... 31
read_zip_to_rasters ............................... 31
run_shiny_animaltracker ....................... 32
run_validation_app .................................. 32
**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**

```r
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**

```r
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 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**

```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  
*Clean 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")
```

**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**

```r
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**

list of cleaned animal GPS datasets

**Examples**

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

---

**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**

```r
clean_location_data(  
  df,  
  dtype,  
  prep = TRUE,  
  filters = TRUE,  
  aniid = NA,  
  gpsid = NA,  
  maxrate = 84,  
  maxcourse = 100,  
  maxdist = 840,  
  maxtime = 60 * 60,  
  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
prep reformat columns if all required columns are not present, defaults to True
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
data frame 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

```r
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 reformat 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

```r
compare_flags(
  correct,
  candidate,
  use_elev = TRUE,
  use_slope = TRUE,
  has_flags = FALSE,
  dropped_flag = NULL
)
```

Arguments

- `correct` reference data frame
- `candidate` df to be compared to the reference
- `use_elev` logical, whether to include elevation in comparison, defaults to true
- `use_slope` logical, whether to include slope in comparison, defaults to true
- `has_flags` logical, whether correct data frame has predefined flags, defaults to false
- `dropped_flag` dropped flag column, must be defined when has_flags is true, otherwise null

Value

joined and reformatted data frame

Examples

```r
# 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**

```r
compare_summarise_daily(
  correct,
  candidate,
  use_elev = TRUE,
  export = FALSE,
  out = NULL
)
```

**Arguments**

- `correct`: reference data frame
- `candidate`: data frame to be compared to the reference
- `use_elev`: logical, whether to include elevation in summary, defaults to true
- `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 data frame

**Examples**

```r
# 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,  # reference data frame
  candidate,  # data frame to be compared to the reference
  use_elev = TRUE,  # logical, whether to include elevation in summary, defaults to True
  export = FALSE,  # logical, whether to export summaries to .csv, defaults to False
  gps_out = NULL,  # desired file name of .csv output summary by GPS collar when export is True
  date_out = NULL  # desired file name of .csv output summary by date when export is True
)
```

Arguments

- `correct`: reference data frame
- `candidate`: data frame to be compared to the reference
- `use_elev`: logical, whether to include elevation in summary, defaults to True
- `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 data frames

Examples

```r
# Compare and summarise unfiltered demo cows to filtered
compare_summarise_data(demo_unfiltered_elev, demo_filtered_elev)
```
**Description**

'shiny' module server-side UI generator for the animaltracker app's date picker.

**Usage**

```r
datePicker(input, output, session, req_list, text)
```

**Arguments**

- `input`: 'shiny' server input, automatically populated
- `output`: 'shiny' server output, automatically populated
- `session`: 'shiny' server session, automatically populated
- `req_list`: list of reactive statements required to display picker
- `text`: title for picker

**Value**

'`shiny`' renderUI object for date picker

---

**Description**

'shiny' module UI output for the animaltracker app's date picker.

**Usage**

```r
datePickerOutput(id)
```

**Arguments**

- `id`: chosen ID of UI output

**Value**

'`shiny`' uiOutput for date picker
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**

Description
---
Demo comparison of two animal datasets

Usage
---
demo_comparison

Format
---
A data frame with 2758 rows and 33 variables

---

demo_filtered

Description
---
Filtered demo animal GPS data from cows

Usage
---
demo_filtered

Format
---
A data frame with 2187 rows and 26 variables

---

demo_filtered_elev

Description
---
Filtered demo animal GPS data from cows with elevation

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**

Unfiltered demo animal GPS data from cows with elevation

**Description**

Unfiltered demo animal GPS data from cows with elevation

**Usage**

demo_unfiltered_elev

**Format**

A data frame with 2288 rows and 35 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 according 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 according 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
**dev_add_to_gitignore**  
*Add big files to a .gitignore file*

**Description**
Add big files to a .gitignore file

**Usage**
```python
dev_add_to_gitignore(data_dir)
```

**Arguments**
- `data_dir`: directory of animal data files

**Value**
None

---

**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**
```python
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
**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**

```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**

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

**Arguments**

- `df`: clean animal data frame
- `file_id`: ID number of source of animal data frame
- `dtype`: igotu or columbus
- `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"))

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

```r
join_summaries(
  correct_summary,
  candidate_summary,
  by_str,
  daily = FALSE,
  use_elev = TRUE
)
```

Arguments

- `correct_summary`: summary data frame of reference dataset, returned by `summarise_anidf`
- `candidate_summary`: summary data frame 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
- `use_elev`: logical, whether to include elevation in summary, defaults to true

Value

data frame of joined summaries with differences

Examples

```r
# 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**

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

**Description**

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

**Usage**

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

**Arguments**

- `correct`: reference data frame
- `candidate`: data frame 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**

```r
# 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**

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**

```r
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

Description

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

Usage

```r
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

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.

Usage

```r
qqplot_time(rds_path)
```
quantile_time

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"))
reactivePicker  
‘shiny’ module server-side UI generator for the animaltracker app’s dynamic dropdown selections.

Description

‘shiny’ module server-side UI generator for the animaltracker app’s dynamic dropdown selections.

Usage

reactivePicker(
  input,
  output,
  session,
  type,
  req_list,
  text,
  min_selected = NULL,
  max_selected = NULL,
  multiple,
  options = NULL
)

Arguments

input  ‘shiny’ server input, automatically populated
output ‘shiny’ server output, automatically populated
session ‘shiny’ server session, automatically populated
type purpose of picker - currently supported types are "site", "ani", and "recent"
req_list list of reactive statements required to display picker
text title for picker
min_selected index of lowest selected value in possible choices, should be null if type is "recent"
max_selected index of highest selected value in possible choices should be null if type is "recent"
multiple logical, whether to allow selecting multiple values
options options for shinyWidgets pickerInput

Value

‘shiny’ renderUI object for dropdown selection
**reactivePickerOutput**

'**shiny**' module UI output for the animaltracker app’s dynamic dropdown selections.

**Description**

'**shiny**' module UI output for the animaltracker app’s dynamic dropdown selections.

**Usage**

reactivePickerOutput(id)

**Arguments**

- **id**: chosen ID of UI output

**Value**

'**shiny**' uiOutput object for dropdown selection

---

**reactivePlot**

'**shiny**' module server-side UI generator for the animaltracker app’s summary statistics tables.

**Description**

'**shiny**' module server-side UI generator for the animaltracker app’s summary statistics tables.

**Usage**

reactivePlot(input, output, session, plot_type, dat)

**Arguments**

- **input**: '**shiny**' server input, automatically populated
- **output**: '**shiny**' server output, automatically populated
- **session**: '**shiny**' server session, automatically populated
- **plot_type**: plot type to generate
- **dat**: animal data frame

**Value**

'**shiny**' renderPlot object
reactivePlotOutput

`shiny` module UI output for the animaltracker app’s plots tab.

Description

`shiny` module UI output for the animaltracker app’s plots tab.

Usage

`reactivePlotOutput(id)`

Arguments

id chosen ID of UI output

Value

`shiny` `plotOutput` object

reactiveRange

`shiny` module server-side UI generator for the animaltracker app’s coordinate range input.

Description

`shiny` module server-side UI generator for the animaltracker app’s coordinate range input.

Usage

`reactiveRange(input, output, session, type, dat)`

Arguments

input `shiny` server input, automatically populated
output `shiny` server output, automatically populated
session `shiny` server session, automatically populated
type latitude or longitude
dat animal data frame

Value

`shiny` `renderUI` object for coordinate range input
reactiveRangeOutput

Example: reactiveRangeOutput(id)

Arguments

- id: chosen ID of UI output

Value

- 'shiny' uiOutput for coordinate range input

read_columbus

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

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

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

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)

**Value**

None

---

run_validation_app

*Run the 'shiny' validation app*

**Description**

Run the 'shiny' validation app

**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

staticPicker

'shiny' module server-side UI generator for the animaltracker app's basic dropdown selections.

Description

'shiny' module server-side UI generator for the animaltracker app's basic dropdown selections.

Usage

staticPicker(
  input,
  output,
  session,
  selected_ani,
  text,
  choices,
  min_selected,
  max_selected
)
staticPickerOutput

Arguments

- **input**  
  "shiny" server input, automatically populated
- **output**  
  "shiny" server output, automatically populated
- **session**  
  "shiny" server session, automatically populated
- **selected_ani**  
  selected animals from animaltracker app input
- **text**  
  title for picker
- **choices**  
  vector of possible choices for picker
- **min_selected**  
  index of lowest selected value in choices
- **max_selected**  
  index of highest selected value in choices

Value

"shiny" renderUI object for dropdown selection

---

staticPickerOutput | Shiny Module UI output for the animaltracker app's basic dropdown selections.

Description

Shiny Module UI output for the animaltracker app’s basic dropdown selections.

Usage

```r
staticPickerOutput(id)
```

Arguments

- **id**  
  chosen ID of UI output

Value

"shiny" uiOutput object for dropdown selection
**stats**  
'shiny' module server-side UI generator for the animaltracker app’s summary statistics tables.

**Description**

'shiny' module server-side UI generator for the animaltracker app’s summary statistics tables.

**Usage**

```r
stats(
  input,  
  output,  
  session,  
  selected_cols,  
  selected_stats,  
  col_name,  
  col,  
  dat  
)
```

**Arguments**

- **input**  
  'shiny' server input, automatically populated
- **output**  
  'shiny' server output, automatically populated
- **session**  
  'shiny' server session, automatically populated
- **selected_cols**  
  selected columns from animaltracker app input
- **selected_stats**  
  selected summary statistics from animaltracker app input
- **col_name**  
  column name to compute summary statistics
- **col**  
  column to compute summary statistics
- **dat**  
  animal data frame containing col

**Value**

'shiny' renderTable object for table
statsLabel

'shiny' module server-side UI generator for the animaltracker app's summary statistics labels.

Description

'shiny' module server-side UI generator for the animaltracker app's summary statistics labels.

Usage

statsLabel(
  input,
  output,
  session,
  selected_cols,
  selected_stats,
  col_name,
  text
)

Arguments

input    'shiny' server input, automatically populated
output   'shiny' server output, automatically populated
session  'shiny' server session, automatically populated
selected_cols selected columns from animaltracker app input
selected_stats selected summary statistics from animaltracker app input
col_name  column name to compute summary statistics
text      text of summary statistics label

Value

'shiny' renderUI object for label

statsLabelOutput

'shiny' Module UI output for the animaltracker app’s summary statistics labels.

Description

'shiny' Module UI output for the animaltracker app’s summary statistics labels.

Usage

statsLabelOutput(id)
```r
statsOutput

Arguments
id           chosen ID of UI output

Value
'shiny' uiOutput object for label

Description
'shiny' module UI output for the animaltracker app’s summary statistics tables.

Usage
statsOutput(id)

Arguments
id           chosen ID of UI output

Value
'shiny' uiOutput object for table

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

```r
summarise_anidf(
  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 = NULL,  # elevation column, must be defined when use_elev is true, otherwise NULL
  use_elev = TRUE,  # logical, whether to include elevation in summary, defaults to true
  daily = FALSE  # whether to group by both GPS and Date for daily summary, defaults to 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, must be defined when use_elev is true, otherwise NULL
- `use_elev`: logical, whether to include elevation in summary, defaults to true
- `daily`: whether to group by both GPS and Date for daily summary, defaults to false

Value

data frame of summary statistics for the animal data frame

Examples

```r
# Summary of demo data by date
summarise_anidf(demo, Date, Latitude, Longitude, Distance, Course, Rate, Elevation)
```
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

```r
# Read in .rds of demo data and summarise by GPS unit
summarise_unit(system.file("extdata", "demo_nov19.rds", package = "animaltracker"))
```

---

**Description**

'shiny' module server-side UI generator for the animaltracker app's time input.

**Usage**

```r
time(input, output, session, type, meta, selected_ani)
```

**Arguments**

- `input`: 'shiny' server input, automatically populated
- `output`: 'shiny' server output, automatically populated
- `session`: 'shiny' server session, automatically populated
- `type`: min or max
- `meta`: animal metadata from app, must be non-empty for time input to display
- `selected_ani`: selected animals from app, must be non-empty for time to display

**Value**

'shiny' renderUI object for time input

---

**Description**

'shiny' module UI output for the animaltracker app's time input

**Usage**

```r
timeOutput(id)
```

**Arguments**

- `id`: chosen ID of UI output
violin_compare

Value

'shiny' uiOutput for time input

Descripción

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

Usage

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

* datasets
  demo, 14
demo_comparison, 15
demo_filtered, 15
demo_filtered_elev, 15
demo_info, 16
demo_meta, 16
demo_unfiltered, 16
demo_unfiltered_elev, 17
  get_meta, 19
  histogram_animal_elevation, 20
  histogram_time, 20
  histogram_time_unit, 21
  join_summaries, 22
  line_compare, 23
  lookup_elevation_aws, 23
  lookup_elevation_file, 24
  process_elevation, 25
  qqplot_time, 25
  quantile_time, 26
  reactivePicker, 27
  reactivePickerOutput, 28
  reactivePlot, 28
  reactivePlotOutput, 29
  reactiveRange, 29
  reactiveRangeOutput, 30
  read_columbus, 30
  read_gps, 31
  read_zip_to_rasters, 31
  run_shiny_animaltracker, 32
  run_validation_app, 32
  save_meta, 33
  staticPicker, 33
  staticPickerOutput, 34
  stats, 35
  statsLabel, 36
  statsLabelOutput, 36
  statsOutput, 37
  store_batch_list, 37
  summarise_anidf, 38
  summarise_col, 39
  summarise_unit, 39

app_server, 3
app_ui, 4
boxplot_altitude, 4
boxplot_time_unit, 5
calc_bearing, 5
clean_batch_df, 6
clean_export_files, 6
clean_location_data, 7
clean_store_batch, 9
compare_flags, 10
compare_summarise_daily, 11
compare_summarise_data, 12
datePicker, 13
datePickerOutput, 13
deg_to_dec, 14
demo, 14
demo_comparison, 15
demo_filtered, 15
demo_filtered_elev, 15
demo_info, 16
demo_meta, 16
demo_unfiltered, 16
demo_unfiltered_elev, 17
detect_peak_modz, 17
dev_add_to_gitignore, 18
get_data_from_meta, 18
get_file_meta, 19
INDEX

time, 40

violin_compare, 41