LobTag2

An R Based Approach to Tagging Data Management

Installation

Enter these lines into the R console to install the package:

```
> Install.packages("devtools")
> devtools::install_github("jakeelement/LobTag2")
```

After installation, load the package:

```
> library(LobTag2)
```

On loading the package, the following help message will print showing the location of this user guide and the template csv files you will use to load your data:

```
> library(LobTag2)
Welcome to LobTag2!  To get started, your User Guide can be found in: C:/Users/ELEMENTG/AppData/L
ocal/R/win-library/4.2/LobTag2/data/user_guide.pdf And your data entry templates in: C:/Users/ELE
MENTG/AppData/Local/R/win-library/4.2/LobTag2/extdata       Good luck with your tagging project!
```

Read this user guide, then begin loading your data!

Package Description

This package is intended as a tool for researchers studying aquatic animal movement using public reporting of tagged animals. The package runs in R and allows the user to call functions which facilitate intuitive inputting and organization of release and recapture data into an SQL based database (local or Oracle), as well as generating plausible paths of animal movement, with the end product being user generated movement maps. Fundamentally, the package provides a standardized method for maintaining data quality in tagging projects and allowing easy access to these data for further investigations.

R

- This package was written for R version 4.2.2. It is assumed that the user has this or a later version of R installed.

Using on Local Computer (Default)

For normal operation, the package uses RSQLite to create and write an SQL based database file on the user's local hard drive (no external network connections required). This file is stored at C:/LOBTAG/LOBTAG.db

Deleting the LOBTAG.db file will delete your entire database, so it is advised that you backup this file after building your database.

Data tables in LOBTAG.db can be accessed, queried and manipulated using SQL code much like any SQL based database (such as Oracle).

Pro tip: If you're working in R and want to interact with your database in ways outside the functionality of this package, RSQLite is a valuable R package for this. For example, for retrieving your releases data:

```
> con <- dbConnect(RSQLite::SQLite(), "C:/LOBTAG/LOBTAG.db")
> query = paste0("SELECT * FROM LBT_RELEASES")
> releases <- RSQLite::dbSendQuery(con, query)
> releases <- RSQLite::fetch(releases)
```

Using with Oracle

This package has the option to be used to build and interact with Oracle database. To use with Oracle, the user must enter their Oracle credentials as arguments when calling functions. To use Oracle with any of the functions in this package, enter the following when calling the function:

- db = “Oracle”
- oracle.user = “*username*”
- oracle.password = “*password*”
- oracle.dbname = “*database/server name*”

```
> upload_releases(db="Oracle", oracle.user = "madeupuser", oracle.password = "madeuppassword",
  oracle.dbname = "madeupdatabasename")
```

Pro tip: If you’re regularly working with Oracle, saving your credentials as the following variables in your R environment will prevent you having to enter them each time you call a function:

```
> oracle.personal.user = "your user name"
> oracle.personal.password = "your password"
> oracle.personal.server = "your server name"
```

Included data files

The following required files are installed with the package and are stored in the user's R library in the "data" and "extdata" folders (files in green should be opened by the user while using the package normally, files in red are used automatically by the package and should only be opened/manipulated by experienced users):

data:

- NS_extent
- depthraster2.tif
- user_guide.pdf

extdata:

- releases_template.csv
- recaptures_template.csv

Mapping Token

The package's mapping functionality uses Mapbox to generate map tiles. To use this functionality, you will need to create an account with Mapbox in order to get a mapping token. This is quick and easy, just go to:

https://account.mapbox.com/auth/signup/

and create an account. Once you log into your account, look under "Access tokens" and you should see your "Default public token" which is a long string of characters. Will need to enter this value later when using the generate_maps() function.

Functions (Overview)

The package is made up of the following functions used to create and manipulate the tagging database, as well as generate movement maps from tag “release” and “recapture” data entered by the user:

upload_releases()

upload_recaptures()

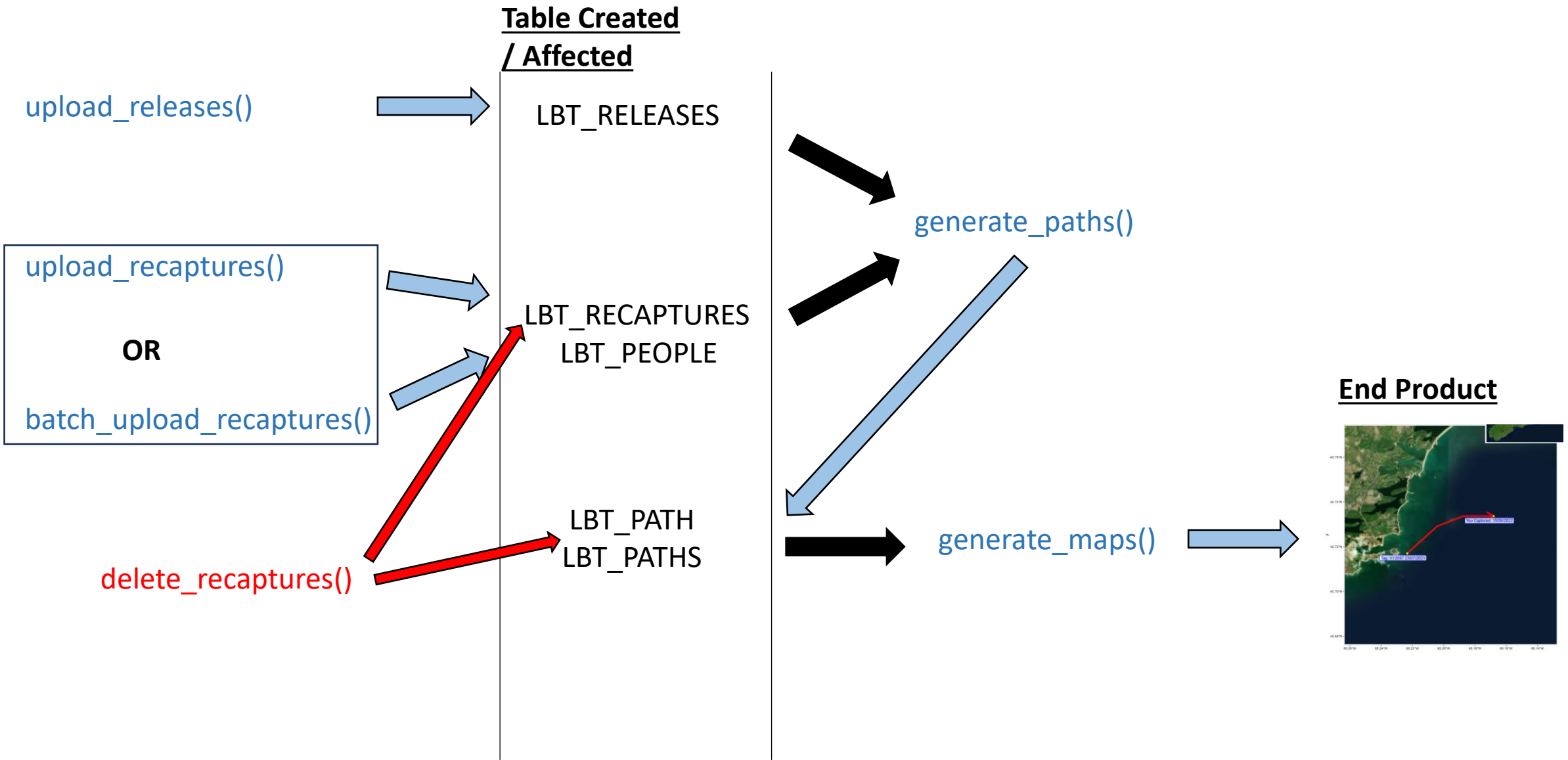
batch_upload_recaptures()

generate_paths()

generate_maps()

delete_recaptures()

Functions (Overview)



Formatting Releases Data:

The first phase of most tagging projects will be releasing the tagged animals. At bare minimum, this will produce data for **location** and **time of release** for each tag. This data will need to be formatted before using with the package.

Use the template file provided to format your release data into a csv for uploading with the package (if you haven't found this template file yet, its location after installing the package can be found by running the following in R):

```
> system.file("extdata", package = "LobTag2")
```

Which will return a path to the folder with your template files, such as:

```
[1] "C:/Users/ELEMENTG/AppData/Local/R/win-library/4.2/LobTag2/extdata"
```

Enter your releases data into the columns provided in the releases_template.csv table. The following columns are mandatory (must contain data for the upload to work):

DAY, MONTH, YEAR, TAG_PREFIX, TAG_NUM, LAT_DEGREES, LAT_MINUTES, LON_DEGREES, LON_MINUTES

The remaining columns are optional and originate from the lobster tagging program for which the package was originally written.

Formatting Release Data: Example

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	SAMPLER	SAMPLER_	AFFILIATI	VESSEL	CAPTAIN	PORT	MANAGEN	DAY	MONTH	YEAR	TAG_COLO	TAG_PREFIX	TAG_NUM	CARAPACI	SEX	SHELL	CLAW	LAT_DEGREES	LAT_MINUTES	LON_DEGREES	LON_MINUTES	COMMENTS
2	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	7	88	3	5		44	35.57	-63	25.7	
3	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	9	106	1	5		44	35.57	-63	25.7	
4	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	10	80	1	5		44	35.57	-63	25.7	
5	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	11	81	1	5		44	35.57	-63	25.7	
6	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	12	75	2	5		44	35.57	-63	25.7	
7	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	13	112	1	5		44	35.57	-63	25.7	
8	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	14	80	1	5		44	35.57	-63	25.7	
9	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	15	94	2	5		44	35.48	-63	25.66	
10	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	16	99	2	5		44	35.48	-63	25.66	
11	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	17	85	1	5		44	35.48	-63	25.66	
12	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	19	127	3	5		44	35.48	-63	25.66	
13	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	20	86	2	5		44	35.48	-63	25.66	
14	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	21	108	3	5		44	35.44	-63	25.66	
15	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	22	83	1	5		44	35.44	-63	25.66	
16	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	23	97	2	5		44	35.44	-63	25.66	
17	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	24	87	2	5		44	35.44	-63	25.66	
18	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	25	100	3	5		44	35.66	-63	25.49	
19	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	26	89	1	5		44	35.66	-63	25.49	
20	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	27	92	3	5		44	35.66	-63	25.49	
21	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	28	103	1	5		44	35.66	-63	25.49	
22	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	29	107	2	5		44	35.66	-63	25.49	
23	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	30	93	2	5		44	35.66	-63	25.49	
24	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	31	69	1	5	1	44	35.66	-63	25.49	
25	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	32	83	2	5		44	35.66	-63	25.49	
26	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	33	97	2	5		44	35.66	-63	25.49	
27	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	34	75	1	5		44	35.66	-63	25.49	
28	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	35	64	1	5		44	35.66	-63	25.49	
29	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	36	85	1	5		44	35.66	-63	25.49	
30	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	37	90	2	5		44	35.66	-63	25.49	
31	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	38	82	3	5		44	35.66	-63	25.49	
32	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	39	93	1	5	1	44	35.66	-63	25.49	
33	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	40	77	1	5		44	35.66	-63	25.49	
34	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	43	78	2	5		44	35.66	-63	25.49	
35	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	44	83	2	5		44	35.66	-63	25.49	
36	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	45	80	1	5		44	35.66	-63	25.49	
37	Ben	Geraint	DFO				33	5	6	2021	Blue	XY	46	79	1	5		44	35.66	-63	25.49	

Green = mandatory columns

Formatting Release Data: Mandatory Columns
TAG_PREFIX

Most tagging programs use a Prefix-Number system to identify unique tags. This package assumes that tag numbers have a prefix to separate them from other tagging programs. This is usually a simple pair of characters (for example, “XY” such as in tag ID “XY1234”) but can be any character combination entered by the user. Even if your tagging program does not use a prefix, one **must** be entered when uploading data with this package. This ensures that the database can hold multiple tagging programs without duplicate tag numbers causing issues.

Formatting Release Data: Mandatory Columns

TAG_NUM

The tag number following the tag prefix is entered as a simple number in its own column called “TAG_NUM”.

Formatting Release Data: Mandatory Columns **Latitude and Longitude**

All release coordinates are uploaded in the format Degrees Decimal Minutes (DDM), split into 4 columns: LAT_DEGREES, LAT_MINUTES, LON_DEGREES, LON_MINUTES. Columns ending in “DEGREES” receive the degree value, and those ending in “MINUTES” receive the MINUTES value as a decimal. For cases when Longitude is west (for example, 58°30.50 W), the proper “-” **must** be included in front of the degree value to denote this, otherwise the package will not know that the longitudes are westerly, for example:

R	S	T	U	V
LAT_DEGREES	LAT_MINUTES	LON_DEGREES	LON_MINUTES	COMM
44	35.57	-63	25.7	
44	35.57	-63	25.7	
44	35.57	-63	25.7	
44	35.57	-63	25.7	
44	35.57	-63	25.7	
44	35.57	-63	25.7	

Release Uploading: Calling the function upload_releases()

Begin the process of uploading release data by calling function upload_releases() which will then prompt you to upload a csv file with the data. By now you should have formatted your release data with the columns found in the provided template file (releases_template.csv).

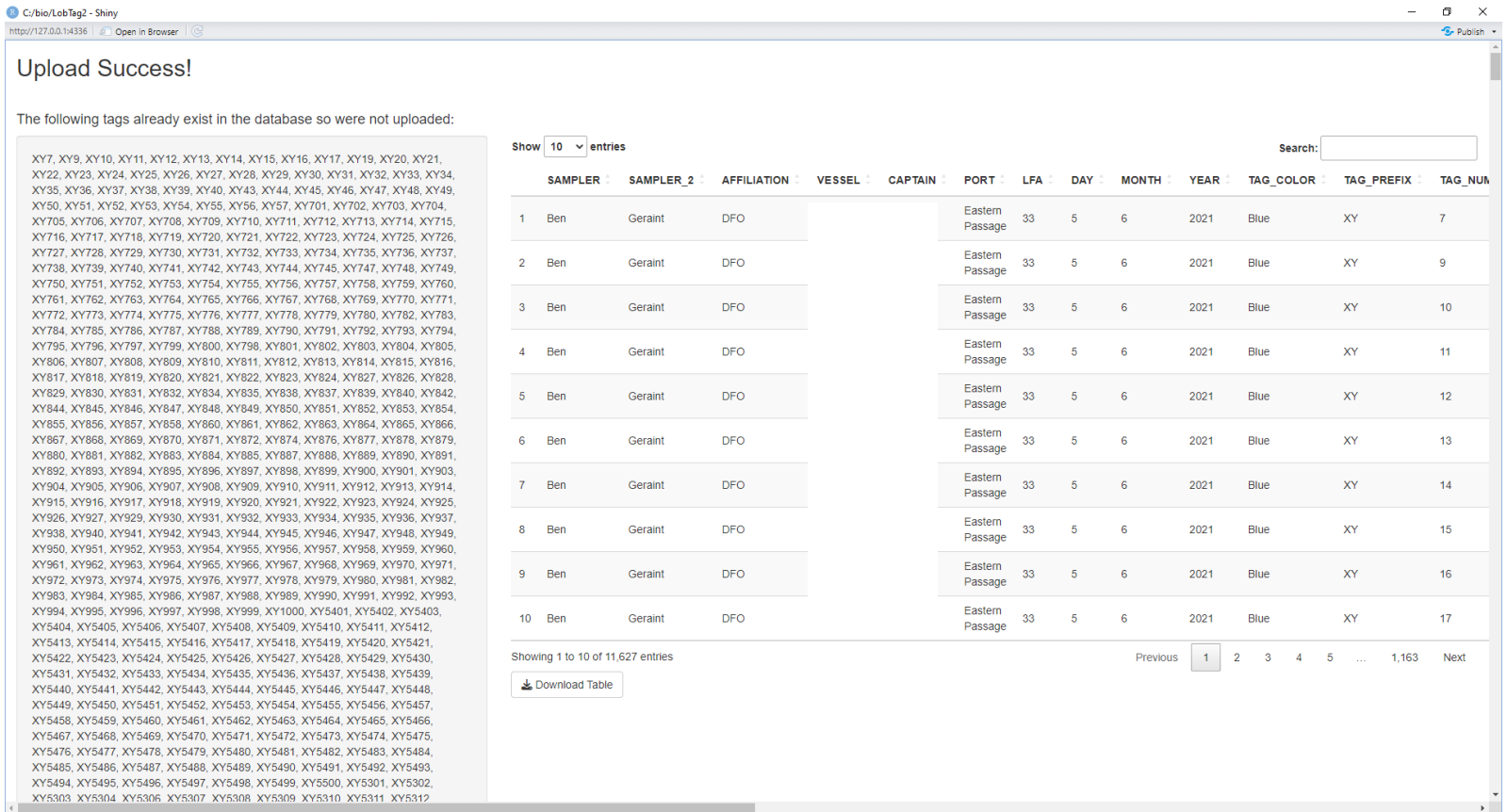
```
> upload_releases()
```

Release Uploading: Error Checking and Upload Completion

When the user has run upload_releases() and selected a data file, the function will perform some general error checking. If there are crucial errors found in the mandatory columns, a dialogue box will appear alerting the user to the location of errors and instructing them to fix these before attempting the upload again. If there are no errors, the user will receive a dialogue alerting them that the upload has completed without errors. This means that the database table LBT_RELEASES has been created and now contains the release data. If there are issues with the data that are concerning but don't prevent uploading, a different dialogue box will appear, warning of these issues and asking the user if they want to proceed with the upload.

Release Uploading: Existing Tags

The function also checks if any of the tags being uploaded are the same as tags already existing in the database, defined as having both identical **prefix and number**. Existing tags will not be uploaded again. If the upload file contains existing tags, the upload will still proceed, but the user will receive a dialogue box alerting them of the tags that were skipped because they were found in LBT_RELEASES.

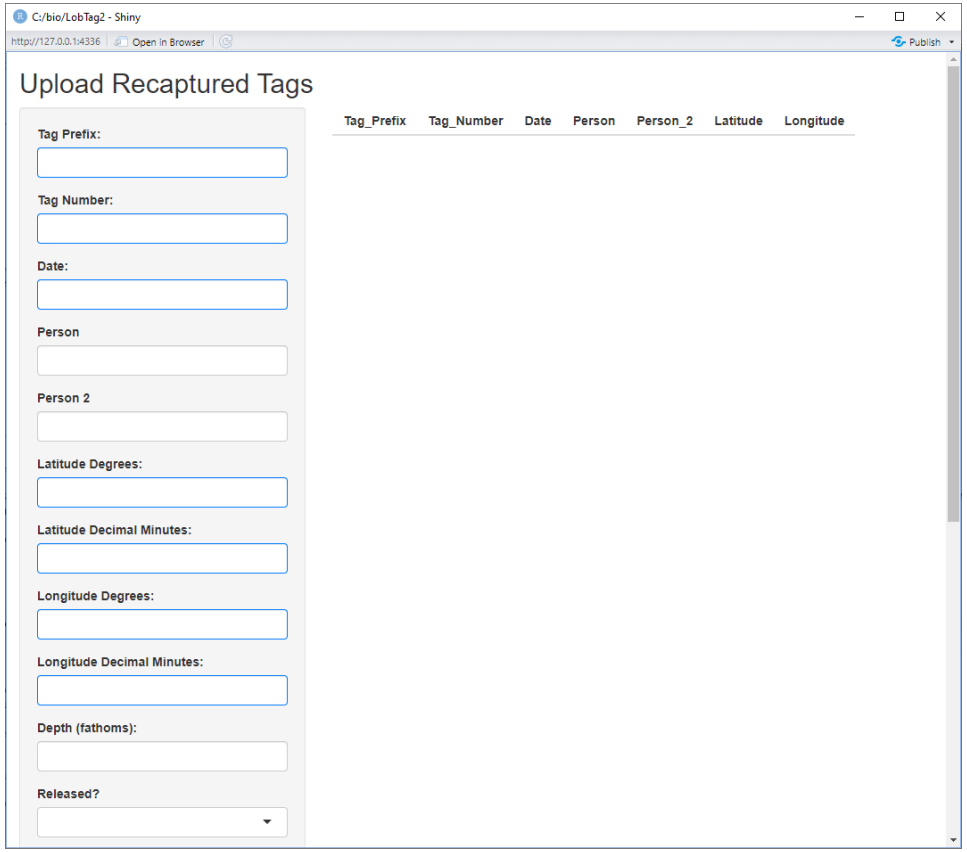


Individual Recapture Uploading:

upload_recaptures()

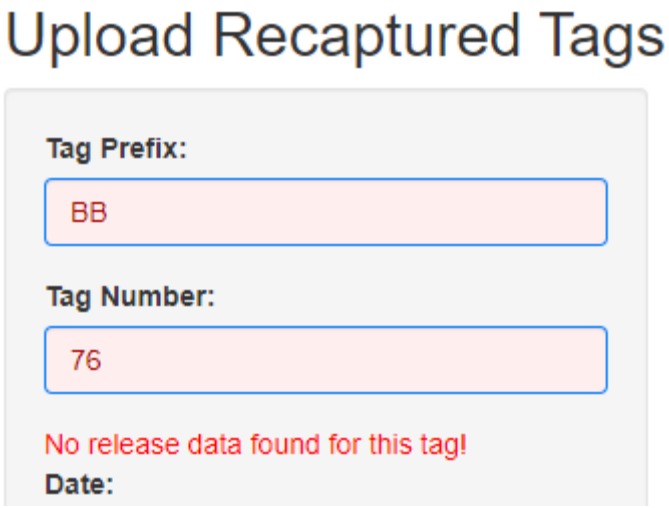
Generally, once tags are released, recapture reports will then come in individually over time. For public programs, these data may be highly variable in format and quality. Running the function upload_recaptures() provides the user with a data entry window which standardizes the entry of these data into the database. Mandatory fields are highlighted in blue.

```
> upload_recaptures()
```



Individual Recapture Uploading: Tag Checking

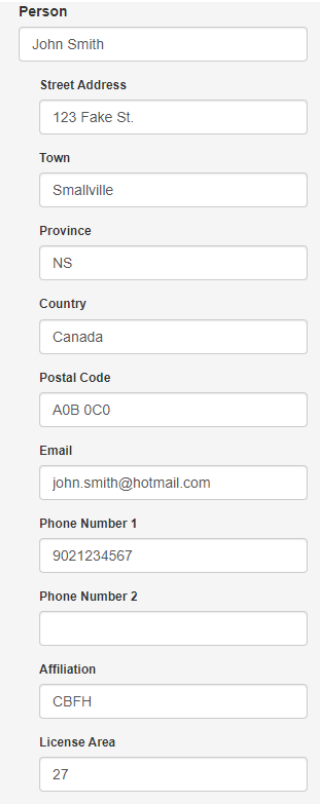
The user interface for uploading recaptures communicates with the database in real-time, allowing it to check if the tag's release data exists in LBT_RELEASES. If the Prefix and Number combination are not found, the user will be alerted:



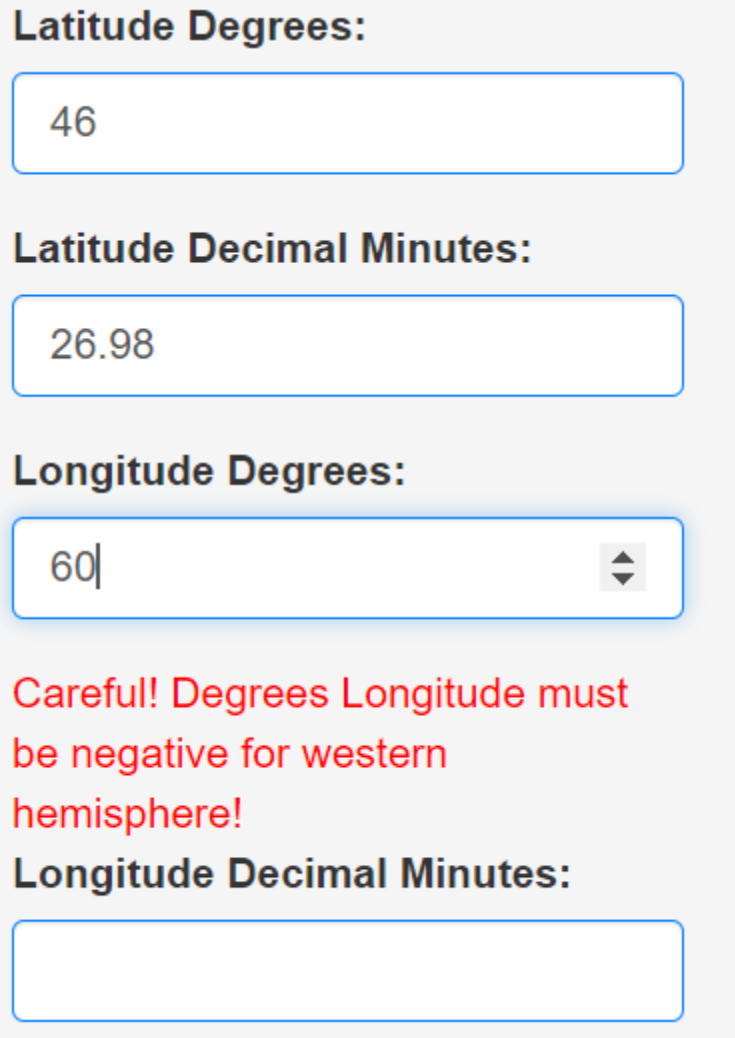
This is just a warning, and the user may still choose to upload this tag. However, if release data are not eventually entered for this tag, then path generation will not be possible.

Individual Recapture Uploading: Person Info

Some tagging programs may wish to collect contact information for the person reporting tag recaptures to allow a reward system to be implemented. For this purpose, the Oracle database includes the LBT_PEOPLE table, which is created/updated when a name is entered in the “PERSON” field during recapture uploading. When this happens, additional fields will appear prompting the user for more contact information. Since the user interface communicates with the database, these will auto-fill with any existing information for the entered name so this does not need to be re-entered each time the person reports a recapture.



Individual Recapture Uploading: Latitude and Longitude

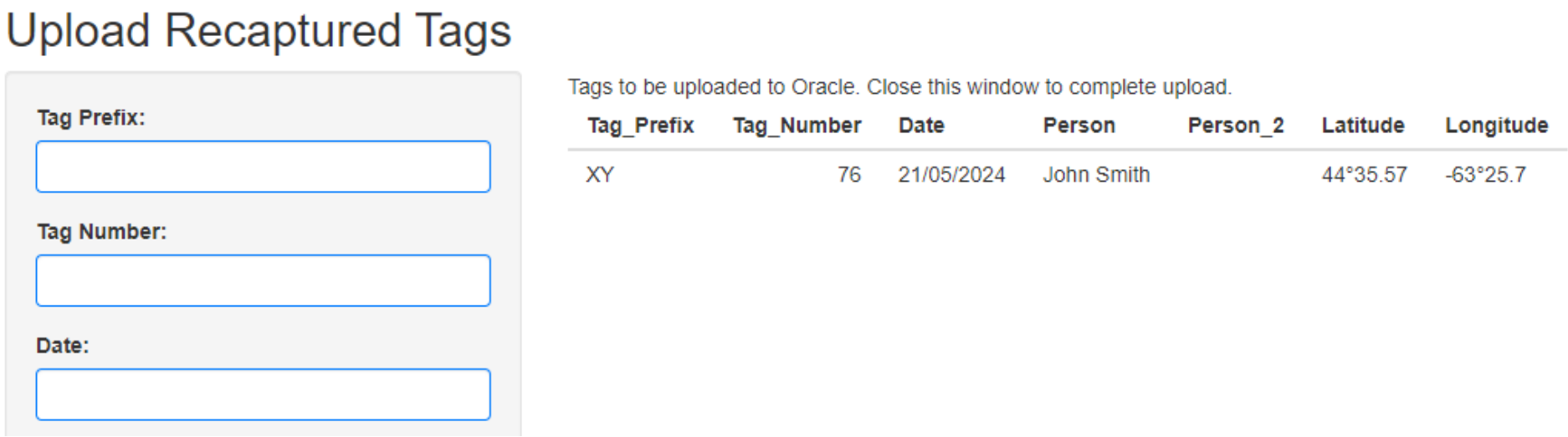


The user interface largely standardizes the input of coordinate values, making it impossible to enter a “non-coordinate”, however, the user must be aware and check that:

- Coordinates are being entered in **DDM** format
- Westerly values for “Longitude Degrees” are **negative**. The scroller for this value goes down for negative values. This is the only way for the functions to know that coordinates are in the western hemisphere. Since the package is likely to be used in the western hemisphere, a warning message will appear if the user enters a positive value here.

Individual Recapture Uploading: Submitting Recaptures

When the user clicks “Submit” in the recapture uploading window, the information for the submitted tag appears in the upper right, and each tag submitted will appear listed this way so that the user can see what tags they are submitting. Upload of all the listed tags will not occur until the user closes the window.



Batch Recapture Uploading: Formatting

If the tagging program already has many recapture data stored in tables, these can be batch uploaded to the database much like the releases data. Like the releases data, these will first need to be formatted using the provided template (recaptures_template.csv). For recapture formatting, the mandatory columns are:

TAG_PREFIX, TAG_NUMER, DAY, MONTH, YEAR,

And **either**: LAT_DEGREE, LAT_MINUTE, LON_DEGREE, LON_MINUTE

Or: LAT_DD, LON_DD (these last two columns allow the user the option to enter coordinates in decimal degree (DD) format if this is the format your coordinates are in. Uploading will work as long one type of coordinate is fully entered).

Batch Recapture Uploading: Formatting Example

	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	
1	DAY	MONTH	YEAR	PERSON	CIVIC	TOWN	PROV	COUNTRY	POST	EMAIL	PHO1	PHO2	AFFILIATI	LICENSE_	PERSON_	LAT_DEGREE	LAT_MINUTE	LON_DEGREE	LON_MINUTE	LAT_DD	LON_DD	FATHOMS	REL
2	21	4	2023	Lloyd Rob	7084 High	Head of C	NS	Canada	B0J 1N0	Lloydrobic	902-497-3563			LFA32	NA	44	33.55	-63	15	44.5577	-63.2037	0	
3	2	6	2023	Brian Sang	68 Fish Pla	Whitehea	NS	Canada	B0H 1T0	dbsangster@hotmail.com			GCIFA	LFA31A	NA					45.2162	-61.1658	0	
4	19	6	2023	Curtis Young											NA					46.32	-60.2455	6.1	
5	3	7	2023	Curtis Young											NA					46	-60.2308	8.1	
6	5	7	2023	Curtis Young											NA					46.3065	-60.2612	2.6	
7	30	6	2023	Charles Murrant											NA					46.079	-59.8428	0	
8	6	7	2023	Charles Murrant											NA					46.1047	-59.8182	0	
9	15	6	2023	Kent Morford											NA					46.1325	-59.8113	9.6	
10	15	6	2023	Ryan Prendergast											NA					46.0833	-59.85	0	
11	27	6	2023	Ryan Prendergast											NA					46.0833	-59.85	0	
12	20	6	2023	Craig MacLeod											NA					46.0082	-59.8888	0	
13	17	6	2023	Scott MacKinnon Sr											NA					46.252	-60.1647	0	
14	17	6	2023	Scott MacKinnon Sr											NA					46.252	60.1647	0	
15	15	6	2023	Rodney Billard											NA					46.2035	59.7417	8	
16	16	6	2023	Rodney Billard											NA					46.1863	-59.7652	10	
17	20	6	2023	Ryan Munden											NA					46.2067	-59.9167	0	
18	20	6	2023	Ryan Munden											NA					46.2067	-59.9167	0	
19	25	6	2023	Terry Billard											NA					46.1917	-59.8517	3	
20	27	6	2023	Stephen MacIntosh											NA					45.7443	-60.21	0	
21	25	6	2023	Carson Paige											NA					46.015	-59.8433	2	
22	27	6	2023	Carson Paige											NA					46.0083	-59.8767	3	
23	3	7	2023	John Adam Rockett											NA					46.2572	-60.069	2	
24	3	7	2023	Austin LeBlanc											NA					46.3438	-60.2988	7.1	
25	19	6	2023	Adam Sharpe											NA					46.0082	-59.6995	0	

Green = Mandatory
Blue = Coordinate option 1
Purple = Coordinate option 2

Batch Recapture Uploading: Calling the function batch_upload_recaptures()

Begin the process of batch uploading recapture data by calling function batch_upload_releases() which will then prompt you to upload a csv file with the data. By now you should have formatted your recapture data with the columns found in the provided template file (recaptures_template.csv).

```
> batch_upload_recaptures()
```

Error checking and warnings: when the file is chosen, error checking will proceed much like with the releases upload. If crucial errors are found, the user will be told to fix these and start over (run batch_upload_recaptures() again). If non-crucial issues are found, the user will be warned and asked if they want to ignore these and proceed with the upload.

Path Generation: generate_paths()

Once the user has completed uploads of releases and at least one recapture, they can begin generating movement paths between these. Running the function generate_paths() will calculate plausible paths for all recaptures found in LBT_RECAPTURES. This function can be run as many times as needed whenever new recaptures are added; recaptures with existing paths will simply be skipped. Pathing relies on functions in the R package “PBSmapping” and uses a “least cost” method in combination with a depth raster map to calculate plausible paths of movement between tag release and each sequential (chronologically) recapture event.

```
> generate_paths()
```

Note: If a new recapture is added from an earlier date than the most recent recapture for that tag, running generate_paths() will simply delete and regenerate an entirely new path for that tag. This means you can always upload new recaptures from any point in time and still regenerate paths to get your most up-to-date paths.

Path Generation: LBT_PATH & LBT_PATHS

Pathing information generated by generate_paths() is stored in two tables, LBT_PATH and LBT_PATHS. LBT_PATH contains the calculated distance between each chronological location of the tag. In this table, the column CID (Capture ID) contains the chronological numbering of each recapture event. LBT_PATHS contains the proposed plausible movement paths of each tag, based on depth. The POS (Position) column contains the sequence for each coordinate making up the plausible path from the last known real location, up to the next known recapture location. Rows representing the recapture event will contain date and name values in the REC_DATE (Recapture Date) and REC_PERSON (Recapture Person) columns.

Note: These tables are intended for reference by the package when generating maps, but they are formatted by functions in the PBSmapping package, and so experienced users may choose to use these tables directly for additional mapping analyses.

Map Generation: generate_maps()

Once paths have been created, these can be presented graphically using the generate_maps() function. This function has a number of additional arguments that change the output. These are:

- **map.token** = Mapping requires a public mapping token for Mapbox.
- **people** = This specifies the recapture person(s) for whom to generate maps. This will cause maps to be generated showing releases and each recapture location (and the plausible connecting path) for each tag ID reported by the chosen person(s). This can be a single name, or a vector of names (Default = **NULL**).
- **all.people** = (Default = **FALSE**), if changed to **all.people = TRUE**, maps for every person who has reported recaptures will be generated.

```
> generate_maps(people = "John Smith", map.token = "pk.dgfAhSgffsgeHGhwhgZd")
> generate_maps(people = c("John Smith", "Jane Doe"), map.token = "Pk.dgfAhSgffsgeHGhwhgZd")
```

Pro tip: You can avoid having to enter your map.token each time by saving it in your R environment as mapbox.token:

```
> mapbox.token = "Pk.dgfAhSgffsgeHGhwhgZd"  -- Just enter this once!
> generate_maps(people = c("John Smith", "Jane Doe"))
```

Deleting Recaptures

Since paths may be generated from multiple recaptures for a single tag, it is important not to directly delete data from LBT_RECAPTURES. If you want to remove a recapture event from the database, you can use the delete_recaptures() function. Running the function will bring up a dialogue box asking the user to search for the tag prefix, number and date of the recapture. If the correct recapture event is found, the user can then click the **Delete** button to remove the recapture. This will delete all information from LBT_RECAPTURES, LBT_PATH and LBT_PATHS for this recapture, and if there are other recaptures for this tag, all paths will be deleted and regenerated.

```
> delete_recaptures()
```