

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 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:

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 Oracle tables, as well as generating plausible paths of animal movement, with the end product being movement maps **tailored to individual participants who report tags**. 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 table:

```
> 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 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, 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:

data:

- NS_extent
- depthraster2.tif
- user_guide.pdf

extdata:

- releases_template.csv
- recaptures_template.csv

Map 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. Copy a paste this and either save it in R as a variable or paste it directly when using the `generate_maps()` function, like so:

```
generate_maps(map_token = "your token" )
```


Functions (Overview)

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

upload_releases()

upload_recaptures()

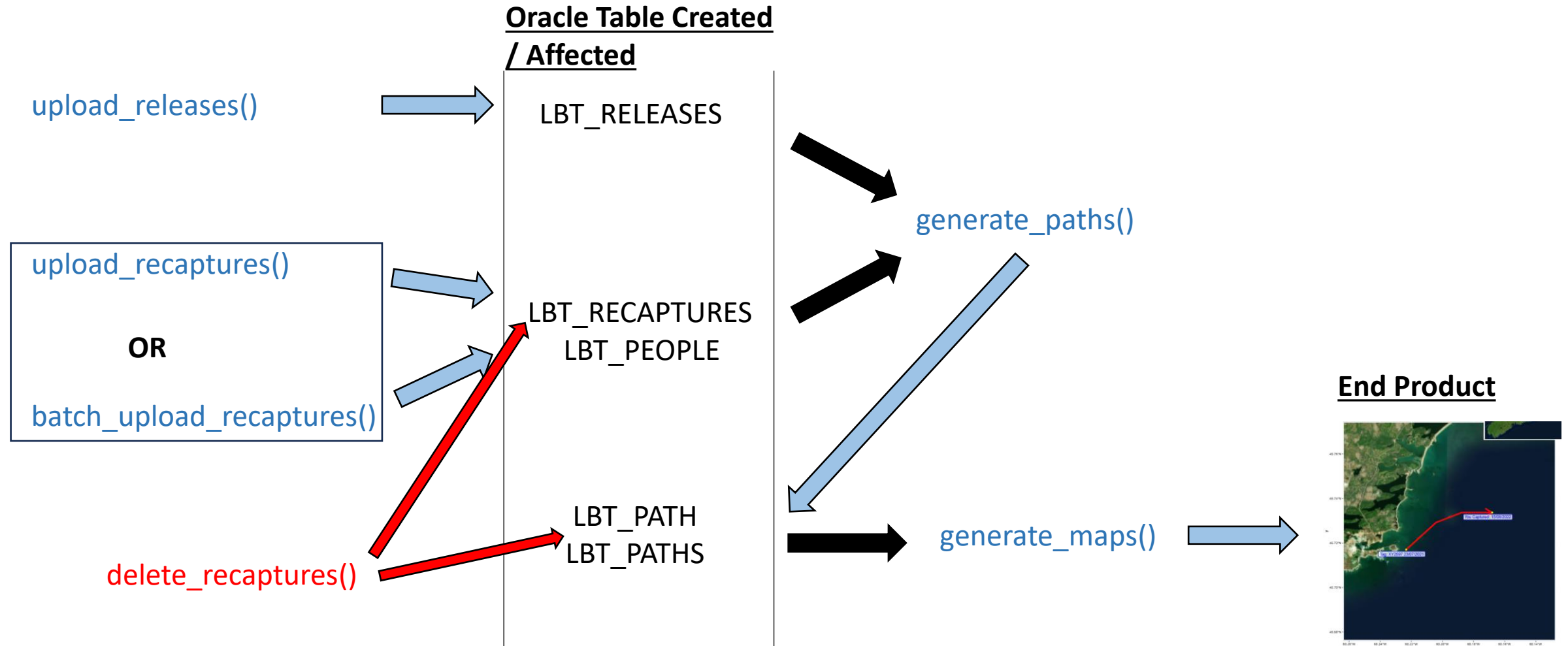
batch_upload_recaptures()

generate_paths()

generate_maps()

delete_recaptures()

Functions (Overview)



Release Uploading:

`upload_releases()`

The first phase of most tagging projects will be releasing the tagged animals. At the bare minimum, this will produce data for location and time of release for each tag. Begin the process of uploading release data by running function `upload_releases()` which will then prompt you to upload a csv file with the data. The package includes a template file (`releases_template.csv`) to standardize creation of this csv file. The following are mandatory columns with specific formatting:

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

The remaining columns are optional and originate from lobster tagging programs.

Release Uploading:

TAG_PREFIX

Most tagging programs use a Prefix-Number system to identify unique tags. The 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 multiple tagging programs can be accommodated without duplicate tag numbers causing issues.

Release Uploading:

TAG_NUM

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

Release Uploading: 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:

[illegible]

Release Uploading: 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	AFFILIATION	VESSEL	CAPTAIN	PORT	MANAGER	DAY	MONTH	YEAR	TAG_COLOR	TAG_PREFIX	TAG_NUM	CARAPACE	SEX	SHELL	CLAW	LAT_DEGREES	LAT_MINUTES	LONG_DEGREES	LONG_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

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.

Release Uploading: Existing Tags

The function also checks if any of the tags being uploaded are the same as tags already existing in the Oracle table, 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.

The following tags already exist in the database so were not uploaded:

SAMPLER	SAMPLER_2	AFFILIATION	VESSEL	CAPTAIN	PORT	LFA	DAY	MONTH	YEAR	TAG_COLOR	TAG_PREFIX	TAG_NUM
1	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	7
2	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	9
3	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	10
4	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	11
5	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	12
6	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	13
7	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	14
8	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	15
9	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	16
10	Ben	Gerant	DFO		Eastern Passage	33	5	6	2021	Blue	XY	17

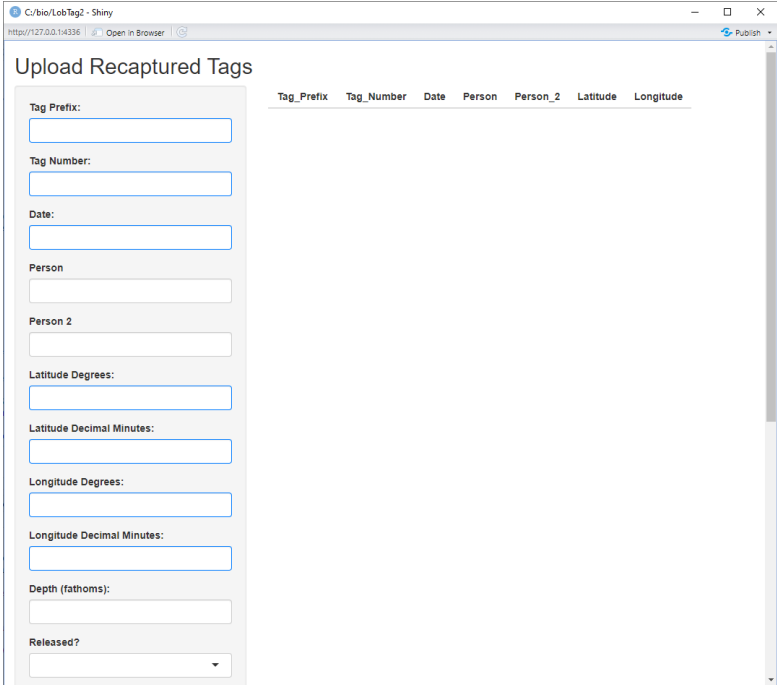
Showing 1 to 10 of 11,627 entries

Previous 1 2 3 4 5 ... 1,163 Next

[Download Table](#)

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.

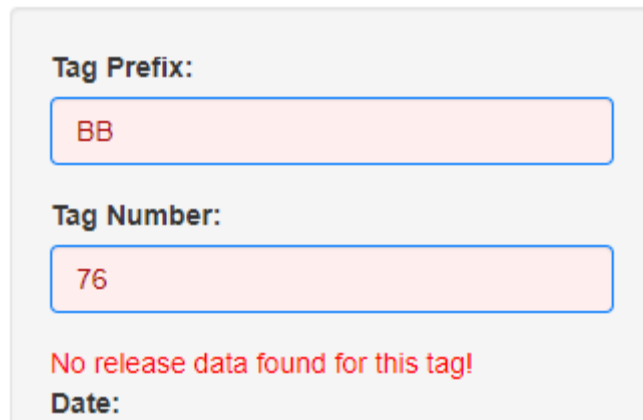


The screenshot shows a web browser window with the title "C:/bio/LobTag2 - Shiny". The address bar shows "http://127.0.0.1:4336". The page title is "Upload Recaptured Tags". Below the title is a table with the following columns: Tag_Prefix, Tag_Number, Date, Person, Person_2, Latitude, and Longitude. To the left of the table is a form with the following fields: Tag Prefix: (text input), Tag Number: (text input), Date: (text input), Person: (text input), Person 2: (text input), Latitude Degrees: (text input), Latitude Decimal Minutes: (text input), Longitude Degrees: (text input), Longitude Decimal Minutes: (text input), Depth (fathoms): (text input), and Released?: (dropdown menu). The fields for Tag Prefix, Tag Number, Date, Person, Person 2, Latitude Degrees, Latitude Decimal Minutes, Longitude Degrees, Longitude Decimal Minutes, and Depth (fathoms) are highlighted in blue, indicating they are mandatory.

Recapture Uploading: Tag Checking

The GUI for uploading recaptures communicates with the Oracle database in real-time, allowing it to check if the tag being entered exists in LBT_RELEASES. If the Prefix and Number combination are not found, the user will be alerted:

Upload Recaptured Tags



The screenshot shows a web form titled "Upload Recaptured Tags". It contains two input fields: "Tag Prefix:" with the value "BB" and "Tag Number:" with the value "76". Below these fields, a red error message states "No release data found for this tag!". At the bottom, there is a "Date:" label followed by an empty input field.

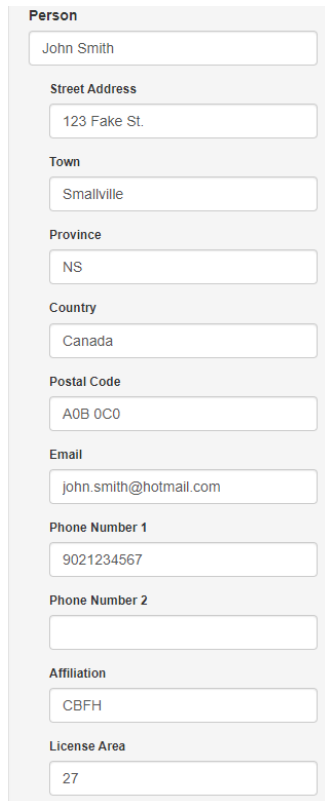
Tag Prefix:	BB
Tag Number:	76
No release data found for this tag!	
Date:	

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

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 GUI communicates with Oracle, these will be auto-filled with any existing information for the entered name so this does not have to be re-entered each time the person reports a recapture.



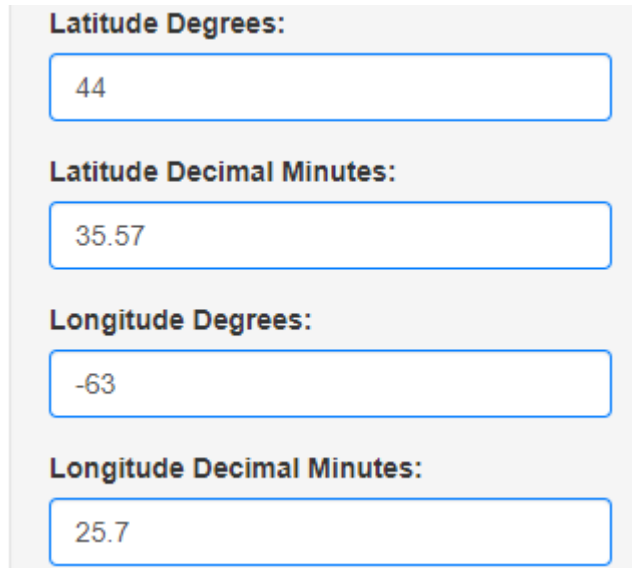
The form is titled "Person" and contains the following fields:

- Person:** John Smith
- Street Address:** 123 Fake St.
- Town:** Smallville
- Province:** NS
- Country:** Canada
- Postal Code:** A0B 0C0
- Email:** john.smith@hotmail.com
- Phone Number 1:** 9021234567
- Phone Number 2:** (empty field)
- Affiliation:** CBFH
- License Area:** 27

Recapture Uploading: Latitude and Longitude

The GUI 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 GUI 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.



The image shows a screenshot of a GUI form with four input fields, each with a label above it. The labels are 'Latitude Degrees:', 'Latitude Decimal Minutes:', 'Longitude Degrees:', and 'Longitude Decimal Minutes:'. The input fields contain the values '44', '35.57', '-63', and '25.7' respectively. The form is styled with a light gray background and blue borders for the input fields.

Field Label	Value
Latitude Degrees:	44
Latitude Decimal Minutes:	35.57
Longitude Degrees:	-63
Longitude Decimal Minutes:	25.7

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 double-check what they just submitted. Upload of all the listed tags will not occur until the user closes the window.

Upload Recaptured Tags

Tag Prefix:

Tag Number:

Date:

Tags to be uploaded to Oracle. Close this window to complete upload.

Tag_Prefix	Tag_Number	Date	Person	Person_2	Latitude	Longitude
XY	76	21/05/2024	John Smith		44°35.57	-63°25.7

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.

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 **straight line 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. The POS (Position) column contains the sequence for each coordinate making up the plausible path from the last real known location, up to the 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.

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.
- “**output.location** = ” Character string specifying the directory in which to put the maps (example: “C:/Users/Username/Documents/”)
- “**person** = ” This specifies the recapture person 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. (Default = NULL)
- “**all.people** = ” (Default = FALSE), if changed to TRUE, maps for every person who has reported recaptures will be generated.