Wildbook R package Design Document

Xinxin Huang

2016/07/19

1. Objective

The primary objective of this package is to interface with the Wildbook mark-recapture ecological database framework. Specially, the functions in this package will allow users to pull data directly via the JDO interface with Wildbook into R and to format the data for further analysis with mark-recapture applications like Program MARK(which can be accessed via the RMark package in R).

Notice that users need to know the JDOQL query language while calling data via the JDO API. With the help of RWildbook package, there is no need for users to learn about JDOQL query language any more and all they need is to specify filters in a R function that we designed in RWildbook. With RWildbook package, we aim to provide convenience to users searching and analyzing data in R. So that users no longer need to go back and forwards from the online database website and the analysis tool.

2. Background

2.1. Wildbook

Wildbook is an always free and open source software which users can use to customize a "wildbook" for one or multiple distinct mark-recapture studies. Based on the Wildlife software framework, a "wildbook" is a website or an online database platform for storing and collecting data about wildlife animals in a study population. Any visitors to a "wildbook" site can contribute the data to the Wildbook framework via the online platform, a "wildbook" site. Within Wildbook, there are collaborations between public data contributors, biologists, bio-statisticians and computer scientists, which save the cost and the time spending on the data collection stage of a wildlife animal research.

There are different types of data structures in Wildbook. For data searching, We will focus on "Encounter" and "Marked Individual" in RWildbook package and the corresponding data constructs within Wildbook are "org.ecocean.Encounter" and "org.ecocean.MarkedIndividual". An encounter is a sighting of an individual animal which represents an individual at a point of time and can be recorded in a photo, a video or a tissue sample of an individual at that time. A marked individual is an individual with an unique identifier which could be the pattern of the appearance, the DNA identifier or a tag. For mark-recapture studies, the first sighting of an individual refers to "mark" and sightings of the same individual afterwards refer to "recaptures". In a "wildbook" site, users can filters data from the online database and customize their study population of one or more species. With these two data structures, Wildbook provides a 21st century way to mark and recapture individuals in front of computer instead of in real life.

2.2. Data Interfaces with Wildbook

Generally, Users can filters data on the wildbook site. Underneath, the website actually gets data from the Wildbook framework via the Wildbook's REST API and JDO API according to users' filters. REST API and JDO API are two main interfaces to search and get data back to users from Wildbook. With the REST and JDO API, users are able to search and get data from Wildbook by calling a URL in certain forms. The URL reflects users' filters.

To show how REST API and JDO API search and get data from Wildbook framework, some example URLs for searching data in Whaleshark.org will be introduced later, where whaleshark.org is one of the

application of Wildbook framework for studies on whales and sharks. The common part of the URL is, http://username:password@whaleshark.org/*.

In the URL, the "*" part can be specified in the following rules according to the api users used and the filters users specified. The rules will be described with some examples.

Some examples of calling data via Wildbook's REST API.

1. For searching a specific marked individual, the search URL is, http://username:password@your_wildbook_URL/rest/org.ecocean.MarkedIndicidual/individualID. For example, the search URL to access the data for a marked individual whose individual ID is "A-001" from *whaleshark.org* is, http://username:password@whaleshark.org/rest/org.ecocean.MarkedIndividual/A-001.
2. For searching a specific Encounter, the search URL is, http://username:password@your_wildbook_URL/rest/org.ecocean.Encounter/EncounterID. For example, the search URL to access the data for a Encounter **6dc75e17-aa8f-4501-a169-4df318b4a815** (a sighting of whale shark A-001) from Wildbook's REST interface in the JSON data format is, http://username:password@whaleshark.org/rest/org.ecocean.Encounter/6dc75e17-aa8f-4501-a169-4df318b4a815.

Calling data via Wildbook's JDO API, the URL is in the form of

http://username:password@your_wildbook_URL/api/jdoql?your_JDOQL_query.

Some examples of the JDOQL query are,

1. For searching a specific marked individual with individual ID "A-001", the JDOQL query in the search URL is, "SELECT FROM org.ecocean.MarkedIndividual WHERE individualID == 'A-001' ".
2. For searching a specific Encounter, the JDOQL query in the search URL is, "SELECT FROM org.ecocean.Encounter WHERE EncounterID == '6dc75e17-aa8f-4501-a169-4df318b4a815' ", where **6dc75e17-aa8f-4501-a169-4df318b4a815** is an encounter ID of a sighting of whale/shark of individual ID "A-001".
3. For searching data with more filters other than just filters the individual ID or the Encounter ID, the Wildbook JDO API will be used. For example, user can call data of all female marked individual by calling the URL with a JDOQL query, "SELECT FROM org.ecocean.MarkedIndividual WHERE sex=='female' ".

When calling data via the JDO API, the URL contains a JDOQL query which describes users' requests on data. JDOQL is a query language which is commonly used in JDO API and the grammar is very similar to SQL query language. Those users who are familiar with JDOQL or SQL can call data directly via the JDO API instead of filtering in the wildbook site.

Also from the examples above, note that the JDO API and REST API return the same data while searching data for a specific encounter or marked individual. Since JDO API provide more flexibility in data searching, the JDO API will be used for data searching in RWildbook package.

2.3. MARK and RMark

Packages `marked` and `RMark` are two famous R packages for Mark-Recapture Analysis. Prior to analyzing the data with the two packages, there will be function to process the raw data set pulled from Wildbook framework in `RWildbook` package.

3. General

There are two main tasks for RWildbook package. One task is to search and pull data from Wildbook into R. The other is to format data for mark-recapture applications like Program MARK. Correspondingly, two main functions will be designed for these two tasks. In this section, we will discuss the ideas of how these two tasks will be accomplished in RWildbook package.

3.1. Collect Data

According to what we discussed in 2.2., the Wildbook's JDO API will be used to call data via the Wildbook framework. Those users who already knows JDOQL and SQL can easily search and get the data by specifying the URL which contains a JDOQL query. With the help of RWildbook, those users who know nothing about JDOQL/SQL can also get data back to R via the Wildbook's JDO API.

Functions in this section are designed for users to filter and pull data via the Wildbook's JDO API. Users can filter the search according to different variables in the Wildbook framework.

Here's some of the variables allow user to filter in RWildbook package:

- individual ID
- encounter ID
- encounter type: approved, rejected, or unapproved
- individual status: alive or dead
- sex: male, female, or unknown
- length measurement
- location
- location ID
- encounter sighting date range
- encounter submission date range
- individual date of birth
- individual date of death
- . . . (More filter variables coming up)

There will be a main function in which some other functions will be used in it. The description table for the five functions can be found below. These three functions will be discussed in more details later.

Table 1: Functions for data searching in RWildbook

Function	Description
searchWB(main)	To pull data via Wildbook's JDO API
WBsearchURL	To generate the search URL
WBjdoql	To generate the JDOQL query according to the filters
filterstring	To generate the JDOQL portion of corresponding filter
sexstring	To generate the JDOQL portion of "sex" filter

3.2. Format Data

As we discussed in Section 2.3, the `markedData` function is to process the data set which is returned by `searchWB` function for the mark-recapture analysis with packages `marked` and `RMark`. Note that packages `marked` and `RMark` use the same format. In `marked` package, the `process.data` function is to process encounter history data frame for **MARK** analysis. Therefore, `markedData` function is to process the raw data set and return the right form of data set for `process.data` function.

The `data` argument is the data frame for `process.data`. The mandatory variable of the `data` is `ch` which contains the capture history of animals. `ch` is a character string which is composed of a constant length sequence of single characters (01001). `data` can contains `freq` variable which is the frequency of the capture history. If `data` is not included, the default structure is `freq`=1. `data` can contains other variables such as sex, location, age and so on. More detail about `process.data` can be found in the help page of `process.data` function.

According to the discussion above, the main task for `markedData` function is to create the `ch` field and abstract the other variables from the raw data set which is returned by `searchWB`.

In order to generate the `ch` field from the encounter information, another important task for `markedData` function is to set the capture time interval. There will be arguments of `markedData` to define the capture period.

In short, the input of `markedData` function is the raw data set from `searchWB` function. `markedData` function returns a list of two R object, one is a data frame which at least contains `ch` and `individualID`, the other is a vector of labels for the capture time intervals. The data frame can also contains the frequency of the capture history and other individual related variables like sex, age location and so on.

4. Functions details

In this section, more function details will be introduced. Especially the objective, argument list and returned result will be discussed in more details for each function in this sections.

4.1. Search and Call Data

Recall that there are three functions for searching and calling data from Wildbook framework shown in Table 1.

4.1.1. searchWB(the main data searching function)

The objective of this function is to pull data via the Wildbook's JDO API. Within `searchWB`, functions `WBjdoql` and `WBsearchURL` will be called automatically to generate the search url according to the values of the arguments and they will be introduced in 3.1.2 and 3.1.3. The arguments of `searchWB` are the user name, the password, the base URL, the type of the Wildbook data, the filters and the JDOQL query which is the combination of the argument lists of functions, `WBjdoql` and `WBsearchURL`. In function `searchWB`, functions `fromJSON` and `url` in {JSONlite} package are used to request data via the Wildbook's JDO API with a search url containing a JDOQL query.

This function will return a data frame which contains the request data.

4.1.2. WBsearchURL

This function is to generate the search URL with a JDOQL query. The arguments of `WBsearchURL` are the user name, the password, the base URL and the JDOQL query. This argument setting allows users to write the JDOQL query on their own or simply use the returned JDOQL query from function `WBjdoql` in section 3.1.2. This function returns the search URL in the class of character. The usage of `WBsearchURL` will mainly be included in the main function, `searchWB`. That means when user call the main function in R, `WBsearchURL` will be automatically called in `searchWB` to create the search URL which will be used to require and pull data via Wildbook's JDO API.

4.1.3. WBjdoql

This function is to generate a JDOQL query according to the filters that the user specified. The arguments of `WBjdoql` are the type of the Wildbook data and all the variables that allow users to specified. This function returns the JDOQL query in the class of character. When calling the data, user can check the help document for the list of variables which allow to filter. The usage of `WBjdoql` will mainly be included in the main function, `searchWB`. That means when user call the main function in R, `WBjdoql` will be automatically called in `searchWB` as a fundamental part of URL which is used to pull data in Wildbook.

4.1.4 filterstring

This function is to generate a character string for a filter variable which is part of the JDOQL query. There will be different parts in the JDOQL query that represent different filter variables. According to the grammar of the JDOQL query, the part represents a filter value is often in the form of "(filtername==filtervalue1)&&(filtername==filtervalue2)". And there will be more complicated form for some of the filter variable. In order to generate the correct string in the JDOQL query, we design this function `filterstring` which provides flexibility in generating the filter string of different forms. The arguments of `filterstring` are the filter name, the filter value, the connection operator and the logical operator. Where the filter name is a character. The filter value, the connection operator and the logical operator can be either a single number/character or a vector. The connection operator is the operator connecting the filter name and the filter values which can be "==", "!=", "<=", ">=", "<", ">". The logical operator is to connect filter string for different value of the same variable which can be "&&" for logical AND or "||" for logical OR. Function `filterstring` returns a character string corresponding to the filter specified in the argument list. This function will be automatically called in function `WBjdoql` to generate the JDOQL portion of the filters.

4.1.5 sexstring

In Wildbook framework, since there are cases that the sex of a marked individual is unknown, there are three values for sex filter which are "male"","female" and "unknown". Therefore when filtering the sex variable in Wildbook framework, the JDOQL query use the negative check language instead of the positive one, e.g. when search all the female individuals, system uses "(!sex.startsWith('male') && !sex.startsWith('unknown') && sex != null)" instead of "sex=='female' " in the JDOQL query. This makes the string for the sex filter complicated and special. We design a function `sexstring` to generate the JDOQL string for sex filter. The argument of `sexstring` is the value of sex filter which can be a vector of any combination of "male"","female" and "unknown". Function `sexstring` returns the JDOQL string according to the value.

Table 2: `sexstring` outcomes for different values

Argument	Outcome
c("male")	(!sex.startsWith('female') && !sex.startsWith('unknown') && sex != null)
c("female")	(!sex.startsWith('male') && !sex.startsWith('unknown') && sex != null)
c("unknown")	(!sex.startsWith('male') &&!sex.startsWith('female'))
c("male","female")	(sex.startsWith('unknown') && sex != null)
c("male","unknown")	(!sex.startsWith('female'))
c("female","unknown")	(!sex.startsWith('male'))
c("male","female","unknown")	NULL

4.1.6 dateTOmillisecond

There are some filter variables related to date which are "sighting date", "encounter submission date", "date of birth" and "date of death". The default format of date value is "year-mm-dd" and the format can be

specified in RWildbook package. The value of date related variable should be a character vector of size two which represents a period of time form one date to another. To filter date variables, instead of directly filtering the date, Wildbook framework filters the millisecond with "1970-01-01" as the origin, e.g., when search for all the encounter sighted between "1970-01-01" to "1970-01-01", the related JDOQL string is " ((dateInMilliseconds >= 0) && (dateInMilliseconds <= 86340000)) ". From the example, we know that the millisecond for the first date is the start of the date while that for the second date is the end of the date. Function `dateTOmillisecond` is to transfer date into millisecond with default origin as"1970-01-01". The argument of function `dateTOmillisecond` is shown as table 3. Function `dateTOmillisecond` returns a numeric vector of size two which will be used as the"filtervalue" in `filterstring` to generate the complete JDOQL filter string for the date related filter variables.

Table 3: Arguments of function `dateTOmillisecond`

Argument	Value
date	A character vector of size two in the same format as specified
origin	A character in the same format as specified
format	A character to set the format for date value

4.2. Format data

After the data is pulled from the Wildbook, the data will be formatted to prepare for the afterward application. We will initially aim at Program MARK. In R, there are two useful packages `marked` and `RMark` for mark-recapture analysis which are written by the same author.

Argument	Usage
data	A data frame which is returned by `searchWB` function
begin.date	A character object which is the date to begin with in from of data_format
end.date	A character object which is the date to end with in form of data_format
breakpoint	A character vector of dates which are the breakpoints for capture time intervals
period	A character vector of the duration of the capture time intervals when the capture time intervals are not continous
date_format	A character object to set the date fromat, "%Y-%m-%d"(default).
origin	A character object to set the origin when change the date to millisecond, "1970-01-01"(default)

`markedData` function returns a list of two R object, one is a data frame which at least contains `ch` and `individualID`, the other is a vector of labels for the capture time intervals. The data frame can also contains the frequency of the capture history and other individual related variables like sex, age location and so on.

More details about the function will be added after the meeting

5. Reference:

1. http://www.wildbook.org
2. http://wildbook.org/doku.php?id=rest
3. http://www.datanucleus.org/products/accessplatform/rest/api.html
4. http://www.datanucleus.org/products/datanucleus/jdo/jdoql.html
5. http://warnercnr.colostate.edu/~gwhite/mark/mark.htm