R for Motus

Tara L. Crewe and Zoe Crysler

Contents

A walk through the use of R for Motus automated radio-telemetry data

Our goal with this online 'handbook' is to show Motus (https://motus.org) users how to use R to import tag detection data for their project or receiver; explore their data through visualizations and summaries; transform their data, e.g., by determining time since sunrise/sunset or magnetic declination; and to run various analytical procedures. This book therefore goes beyond simply highlighting the functionalities of the Motus R package - it also shows users, who may not be experts in R, to use additional R packages to do basic summaries and plots. We hope the contents will be of use, and if you have suggestions for additional examples, please let us know by emailing motus@birdscanada.org.

Chapter 1

Introduction

The Motus Wildlife Tracking System ('Motus'; Taylor et al. 2017; https://www.motus.org) is an international, collaborative automated radio-telemetry network that is used to track the movement and behaviour of primarily small flying organisms affixed with digitally encoded radio-transmitters. Motus has its roots in the SensorGnome network piloted in 2012-2013. In 2014, a major infrastructure expansion was made possible through a Canada Foundation for Innovation grant to Western University, The University of Guelph, and Acadia University. Since then, Motus has continued to grow through the collaboration of independent researchers and organizations https://motus.org/about/, and is now managed as a program of Bird Studies Canada (https://www.birdscanada.org) in partnership with Acadia University.

Motus is unique among automated telemetry arrays in that all researchers in a geographic region (e.g., the Americas or Europe) use a shared radio frequency. This allows tagged animals to be detected by any receiving station across the network, greatly broadening the spatial scope of potential research questions. Motus users also use a shared data infrastructure and web portal: all data collected from across the network are centrally stored and archived, which allows users to access detections of their tags by anyone's receiver in the network, and individuals that maintain receivers have access to all detections of anyone's tags on those receivers.

Having a shared data infrastructure also means that users can benefit from R functions written specifically for Motus data by any and all users. A Motus R package is currently in development, and the intent of this online 'handbook' is to help Motus users use R to retrieve tag detections, explore, visualize, transform, and analyze Motus data. In it's current form, this book will show you how to import, view and summarize your data. The content of the handbook will continue to evolve and grow as we develop the R package code and along with the analytical needs of the network. Those interested in contributing code to the Motus R package or this handbook can send proposed additions to Tara Crewe (tcrewe@birdscanada.org) or Zoe Crysler (zcrysler@birdscanada.org).

Taylor, P. D., T. L. Crewe, S. A. Mackenzie, D. Lepage, Y. Aubry, Z. Crysler, G. Finney, C. M. Francis, C. G. Guglielmo, D. J. Hamilton, R. L. Holberton, P. H. Loring, G. W. Mitchell, D. R. Noriis, J. Paquet, R. A. Ronconi, J. Smetzer, P. A. Smith, L. J. Welch, and B. K. Woodworth. 2017. The Motus Wildlife Tracking System: a collaborative research network to enhance the understanding of wildlife movement. Avian Conservation and Ecology 12(1):8. https://doi.org/10.5751/ACE-00953-120108.

1.1 What this book does not cover

This book does not currently cover how to register radio tags with Motus, manage tags and station deployments, upload data, and so on. Information to guide you through those tasks can be found under the 'resources' tab on the Motus website at https://motus.org/resources/. Please remember to register your tags and enter tag and station metadata with Motus prior to deploying your tags. Please also see https://motus.org/policy/ to review our collaboration policy and tag registration and fee schedule.

1.2 Prerequisites

This book assumes that you have a basic understanding of R. If you are new to R, we highly recommend 'R for Data Science' by Garrett Grolemund and Hadley Wickham (http://r4ds.had.co.nz/), which covers how to import, visualize, and summarize data in R using the Tidyverse collection of R packages https://www.tidyverse.org/. It also provides useful tips for organizing your workflow to create clean, reproducible code (http://r4ds.had.co.nz/workflow-projects.html). We follow their lead by using Tidyverse throughout this book, and where possible within the package functions.

1.3 How this book is organized

Each section of this book will begin with a paragraph or figure describing the broader intention or outcome of the section, followed by details on how to get there using R. For example, the section on accessing and downloading tag detection data begins with a screenshot of the structure of a dataframe, followed by the R code required to access the data.

1.4 Acknowledgements

Some of the text included in this book was adapted from John Brzustowski's earlier work.

Motus was conceived as the SensorGnome network by Philip Taylor and John Brzustowski at Acadia University. Initial expansion of the network was supported by a Canada Foundation for Innovation Grant to Western University (Christopher Guglielmo), The University of Guelph (D. Ryan Norris), and Acadia University (Philip Taylor). The development of the Motus web interface, R package, and accompanying handbook were made possible through a Canarie grant to Bird Studies Canada (https://www.canarie.ca/). Motus continues to grow as a program of Bird Studies Canada, through the collaboration of numerous independent researchers, organizations, and individuals. A non-exhaustive list of Motus partners and collaborators can be found at https://motus.org/data/partners.jsp. If your organization is not listed, please contact motus@birdscanada.org.

Many people have worked together to bring Motus technology, the web interface, and R-package together. The core 'Motus Team' includes John Brzustowski, Zoe Crysler, Tara Crewe, Jeremy Hussell, Catherine Jardine, Denis Lepage, Stuart Mackenzie, Paul Morrill, and Philip Taylor.

Chapter 2

Loading Required Packages

Please note that some functionalities of the devtools package may require updated versions of R and RStudio. To avoid errors, please ensure you are using the most recent releases of R (https://www.r-project.org/) and RStudio (https://www.rstudio.com/products/RStudio/), and update your R packages using update.packages() in the R console.

```r
update.packages()                           ## to update your existing R packages
```

```r
install.packages("devtools")                ## if you haven't already done this
library(devtools)                           ## load devtools
```

Throughout the book, we use Tidyverse https://www.tidyverse.org/, which is a collection of R packages, including tidyr() and dplyr(). These can be installed with:

```r
install.packages("tidyverse")               ## again, you only need to install once.
library(tidyverse)
```

More information on Tidyverse can be found at https://www.tidyverse.org/, or by browsing 'R for Data Science' by Garrett Grolemund and Hadley Wickham: http://r4ds.had.co.nz/.

2.1 Internal data processing

As an animal moves within the detection range of a Motus station, radio transmissions, or 'bursts', are detected by antenna(s) and recorded by a receiver. These raw detection data are either uploaded to the Motus database instantaneously via internet connection, or downloaded from the receiver and uploaded to Motus manually. Behind the scenes, various functions read and process the raw detections data to produce the tag detections file that users access using the R package (see 'Data Import' below). While most users will not need to call on the internal data processing functions, a complete list of functions within the Motus server R package can be found on GitHub: https://github.com/jbrzusto/motusServer. The code behind each function can be viewed on GitHub, or by typing the following in the R console after loading the R package, replacing 'function.name' with the name of the R function of interest:

```r
function.name()
```

Chapter 3

Accessing and understanding detections data

It is important to note that the success of the Motus network is dependent on the timely upload of detection data from receivers, and on the maintenance of accurate and up to date tag and receiver metadata by collaborators. Users are encouraged to check for updated detection data and metadata each time they run an analysis, because collaborators can add detection data and metadata at any time, and these could influence the completeness of your own detections data.

3.1 Detection data

The static files available for download from https://motus.org/data/downloads contains all detections of your registered tags from across the motus network. A tag project database has a name like project-NNN.motus, where the NNN is the motus project ID.

Eventually, the R package under development will allow you to download all detections of any registered tags from a single receiver that is registered to your project.

3.2 Importing tag detections

Your project's .motus file is available for download at https://motus.org/data/downloads under your project profile. Once downloaded, **make sure that you have the appropriate packages installed and loaded in R as outlined in Chapter 2-LoadingPackage**. Once packages are loaded, you can import your tag detections into R using the following code:

```
file.name <- "C:/data/project-123.motus"    ## replace with the full location of your project data
t <- dplyr::src_sqlite(file.name)
```

3.3 Data structure

Your tag database is stored as an SQLite ('dplyr::src_sqlite') file with the extension '.motus'. The sqlite format was chosen because:

1. it is flexible, allowing for many data formats.
2. it is accessible from many software platforms (not just R).

3. it is **appendable**: the database can be created and updated on disk without having to read in and resave the entire contents. This will save time and computer memory when searching to see if any new detections are available for your project or receiver.

The .motus file contains a series of interelated tables where data are stored in a condensed format to save memory. The following tables are included in the .motus file; a complete list of parameters stored in each table can be found in Appendix Table A1:

1. hits: detections data at the level of individual hits.
2. runs: detections data associated with a run (continuous detections of a unique tag on a given receiver).
3. batches: detections data for a given receiver and boot number.
4. tags: metadata related to tags, e.g., unique identifier, tag characteristics (e.g., burst interval).
5. tagDep: metadata related to tag deployments, e.g., deployment date, location, and species.
6. recvDeps: metadata related to receiver deployments, e.g., deployment date, location, receiver characteristics.
7. antDeps: metadata related to antenna deployments, e.g., deployment height, angle, antenna type.
8. species: metadata related to species, e.g., unique identifier, scientific name, common name.
9. projs: metadata related to projects, e.g., project name, principal investigator.
10. gps: metadata related to gps position of receiver.

Because the file is a dplyr::src_sqlite file, all of the dplyr functions can be used to filter and summarize your .motus database, without needing to first save the data as a *flat* file, i.e., a typical two-dimensional dataframe with every record for each field filled in.

Within the .motus database, the *virtual* table 'alltags' contains everything most users will need, and can be accessed using the dplyr tbl() function:

```r
df <- tbl(t, "alltags")
```

If you look at the underlying structure of the data using str(), you will see that df is a list of length 2:

```r
str(df)
```

The first part of the list, 'src', is a list that provides details of the SQLiteConnection, which includes information on where the database is stored. The second part of the list, 'ops', is also a list, and includes the names of the variables included in the 'alltags' table. In other words, in its current form, the R object 'df' does not hold the data itself (i.e., it is a *virtual* table); rather, it includes the database structure and information required to connect to the underlying data in the .motus file. As stated above, the advantage of storing the data in this way is that it saves memory when accessing very large databases, and the dplyr package can be used to manipulate and summarize the 'alltags' table before collecting the results into a typical "flat" format dataframe.

The following table lists the parameters available in the 'alltags' table, along with a description of each parameter.

3.3.1 Convert the virtual 'alltags' table to a flat dataframe

If you want your entire dataframe in a typical "flat" format, i.e., with every record for each field filled in, you can use the collect() function. The output can then be used to generate a .rds or .csv file of your data, but we caution that producing such a table using the full suite of fields can chew up a lot of memory, and can slow R down considerably when dealing with large datasets:

```r
df.flat <- df %>% collect %>% as.data.frame      ## for all fields in the df
```

```r
head(df.flat)      ## Look at first 6 rows of your df
summary(df.flat)   ## summary of each column in your df
str(df.flat)       ## Look at the structure of your data fields
names(df.flat)     ## field names
```

Table 3.1: Description of fields in the tag detections database

Field	Description
hitID	unique Motus ID for this tag detection
runID	unique Motus ID for the run this detection belongs to
batchID	unique Motus ID for the processing batch this detection came from
ts	timestamp, in seconds since 1 Jan, 1970 GMT; precision: 0.1 ms (SG); 2.5 ms (Lotek).
sig	signal strength in "native units"; for SG: dB (max) (logarithmic, 0 = max possible, -10 = 0.1 * max, etc.
sigSD	std. dev. in signal strength among pulses in this burst. SG Only; NA for Lotek
noise	estimate of background radio noise when tag detected, in dB (max) for SG; NA for Lotek
freq	frequency offset from antenna listening frequency, in kHz for SG only; NA for Lote
freqSD	std. dev. of freq offset among pulses in this burst, in kHz. Values larger than 0.1 kHz suggest a bogus de
slop	total absolute difference (milliseconds) in inter-pulse intervals for this burst between registration and det
burstSlop	signed difference (seconds) between detection and registration burst intervals. This is always 0 for the fi
done	logical: is run finished?
motusTagID	Motus tag ID - unique to each individual tag registered
ant	antenna code (for sensorgnomes, USB port number; for Lotek, whatever they use, usually numeric, but c
runLen	number of tag bursts in the current run; constant for all records having the same runID
bootnum	boot session of receiver for SG; NA for Lotek
tagProjectID	ID of the project that manages this tag.
id	manufacturer ID
tagType	all rows are "ID"??
codeSet	for coded tags, the name of the codeset (e.g. 'Lotek-3')
mfg	tag manufacturer
tagModel	manufacturer's model name for a tag (e.g. 'NTQB-3-2')
tagLifespan	estimated lifespan of tag (days)
nomFreq	nominal tag frequency (MOTUS: Nominal frequency receiver was tuned to, in Mhz. This really only app
tagBI	burst interval of tag, in seconds (e.g., 5.8984)
pulseLen	tag pulse length (milliseconds), if applicable. This value is only assigned based on the sample recording c
speciesID	unique numeric Motus ID (integer) for the species on which the tag was deployed
markerNumber	number for any additional marker placed on organism (e.g., bird band #)
markerType	type of additional marker (e.g. metal band)
depLat	latitude of tag deployment, in decimal degrees N - negative values for Southern hemisphere
depLon	longitude of tag deployment, in decimal degrees E - negative values for Western hemisphere
depAlt	altitude of tag deployment, in meters ASL
comments	additional comments or unclassified metadata for tag (often in JSON format)
startCode	integer code giving method for determining tag deployment start timestamp
endCode	integer code giving method for determining tag deployment end timestamp
fullID	full tag ID as PROJECT#MFGID:BI@NOMFREQ. Not necessarily unique over time. See motusTagID
recv	serial number of the receiver; e.g., SG-1234BBBK5678 or Lotek-12345
site	name assigned to a site by the project manager (e.g. location name). This field is optional, and the same
isMobile	logical; whether the sensor is deployed on a mobile platform (eg. a ship)
projID	unique (numeric) ID of the project that deployed this receiver (e.g., 8)
antType	character; antenna type, e.g. "9-element Yagi", "Omni"
antBearing	numeric; compass direction antenna main axis is pointing at (degrees clockwise from local magnetic Nort
antHeight	numeric; height (meters) of antenna main axis above ground
cableLen	numeric; length of coax cable (meters) connecting antenna to funcubedongle radio (optional)
cableType	type of coax cable connecting antenna to funcubedongle radio (optional)
mountDistance	numeric; distance (meters) between antenna mounting (axis centre) and receiver GPS; SG only
mountBearing	numeric; bearing of base of antenna mounting from receiver GPS (degrees clockwise from magnetic north
polarization1	numeric; antenna polarization angle: azimuth component (degrees clockwise from local magnetic north)
polarization2	numeric; antenna polarization angle: elevation component (degrees above horizon)
spEN	species English (common) name
spFR	species French (common) name
spSci	species scientific name
spGroup	species group, e.g., BIRDS, BATS
tagProj	short label of project that deployed the tag, e.g., "HolbSESA"
projID	short label of project that deployed the receiver

If you want to load only part of your entire data frame (eg. certain fields, only certain tags, all tags from a specified project, species, etc.), you can use dplyr funtions to filter results before collecting the data into a flat dataframe. Some examples are below:

1. To select certain fields;

```
df.flat.subset <- select(df, recv, ant, motusTagID) %>% distinct %>% collect %>% as.data.frame      ## 
```

2. To select certain tag IDs;

```
df.flat.subset <- filter(df, motusTagID %in% c(9939, 25643)) %>% collect %>% as.data.frame    ## filter
```

3. To select a specified species;

```
df.flat.subset <- filter(df, speciesID == 15580) %>% collect %>% as.data.frame    ## filter to only inc
df.flat.subset <- filter(df, spEN == "Swainson's Thrush") %>% collect %>% as.data.frame    ## filter to
```

You can also summarize your data before converting to a flat file. For example, to find the first hourly detection of each tag in each hour by receiver and antenna, you could do this:

```
hourly <- df %>% mutate (hour = 3600 * round(ts / 3600, 0)) %>% distinct (recv, ant, motusTagID, hour)

## and collect these into a data.frame

hh <- hourly %>% collect %>% as.data.frame
```

3.3.2 Export a "flat" dataframe and save as a .csv on your computer

The below example is how you would save the flat file called "df.flat" within R, as a .csv file called "motus_detecitons.csv" on your computer in the location C:/data.

```
write.csv(df.flat, "C:/data/motus_detections.csv")  ## "df.flat"" is the name of the flat file within R
## "C:/data/" is the the location where you would like your .csv stored
## "motus_detections.csv", is the name of the .csv file you are creating, this can be anything you like
```