Data requirements and loading methods for the R package *glatos*

Christopher Holbrook

2018-10-20

Contents

Introduction

This vignette describes minimum data requirements and loading methods for the R package *glatos* (version 0.2.6 and earlier). There are no specific data requirements of the package as a whole. Instead, data requirements are specific to each individual function as described in each function-specific help file (e.g., `?summarize_detections`). To ensure that data from GLATOS and other known sources are loaded efficiently and consistently, the following functions faciliate loading of data from specific sources with standardized data formats.

Data loading functions :

read_glatos_detections reads detection data from a comma-separated-values text file obtained from the GLATOS Data Portal and returns an object of class *glatos_detections*.

read_otn_detections reads detection data from a comma-separated-values text file obtained from the Ocean Tracking Network and returns an object of class *glatos_detections*.

read_glatos_receivers reads receiver data from a comma-separated-values text file obtained from the GLATOS Data Portal and returns an object of class *glatos_receivers*.

read_glatos_workbook reads data from a GLATOS project-specific MS Excel workbook (*.xlsm file) and returns a list of class *class_workbook* with two-elements; one of class *glatos_receivers* and one of class *glatos_animals*.

read_vemco_tag_specs reads tag specification data from an MS Excel workbook (*.xls file) provided by VEMCO and returns a list with two elements containing tag specifications and the tag operating schedule.

Although most of the functions listed above produce objects of *glatos*-specific classes, no *glatos* function will require those class labels. Such classes should merely be thought of as labels showing that the objects were produced by a *glatos* function and therefore the object will be compatible with relevant *glatos* functions, as long as the class labels are not assigned by some other process.

Most functions in the *glatos* package have been designed to accept data in the format returned by the *glatos* load functions, so using the *glatos* load functions will generally ensure that the resulting data conform to the requirements of other functions in the package. If *glatos* load functions are not used, then users will need to ensure that their data meet the requirements of each function used.

This vignette will describe key elements of the data objects mentioned above and show methods for loading data from other sources.

Detection data

Requirements

glatos functions that accept detection data as input will typically require a *data.frame* with one or more of the following columns, named and defined exactly as described below:

- **detection_timestamp_utc** *A* POSIXct *object with detection timestamps (e.g.,* "2012-04-29 01:48:37"*).*
- **receiver_sn** *A character vector with unique receiver identifier. This is needed to associate each record with a specific receiver (a physical instrument).*
- **deploy_lat** *A numeric vector with latitude (decimal degrees, NAD83) of geographic location where receiver was deployed. Must be negative for locations in the southern hemishere and positive for locations in the northern hemisphere (e.g.,* 43.39165*).*
- **deploy_long** *A numeric vector with longitude (decimal degrees, NAD83) of geographic location where receiver was deployed. Must be negative for locations in the western hemishere and positive for locations in the eastern hemisphere (e.g.,* -83.99264*).*
- **transmitter_codespace** *A character string with transmitter code space (e.g.,* "A69-1061" *for Vemco PPM coding). In combination with* transmiter_id *, this is needed to associate each record with a specific transmitter (a physical instrument).*
- **transmitter_id** *A character string with transmitter ID code (e.g.,* "1363" *for Vemco PPM coding). In combination with* transmitter_codespace*, this is needed to associate each record with a specific transmitter (a physical instrument).*
- **sensor_value** *A numeric sensor measurement (e.g., an integer for 'raw' Vemco sensor tags).*
- **sensor_unit** *A character string with sensor_value units (e.g.,* "ADC"* for 'raw' Vemco sensor tag detections).*

Additionally, some functions will require columns to associate each record with an individual animal (or group of animals) or geographic location (or group of locations). These can be specified by the user, but examples of such columns in a GLATOS standard detection file are:

- **Examples of columns that identify individual animals (or groups)**
  - *animal_id*
  - *release_location*
- **Examples of columns that identify receiver locations (or groups)**
  - *glatos_array*
  - *station*
  - *glatos_project_receiver*

Any *data.frame* that contains the above columns should be compatible with all *glatos* functions that accept detection data as input. Use of the data loading functions *read_glatos_detections* and *read_otn_detections* will ensure that these columns are present, but can only be used on data in GLATOS and OTN formats. Data in other formats will need to be loaded using other functions (e.g., *read.csv()*, *data.table::fread()*, etc.) and compatibility with *glatos* functions will need to be carefully checked.

Detections obtained from the GLATOS Data Portal

The *read_glatos_detections()* function reads in detection data from standard detection exports (*.csv files) obtained from the GLATOS Data Portal and checks that the data meet requirements of *glatos* functions. Data are read using *fread* in the *data.table* package, timestamps are formatted as class *POSIXct* and dates are formatted as class *Date*.

First, we will use *system.file()* to get the path to the walleye_detections.csv file included in the `glatos` package.

```
# Set path to walleye_detections.csv example dataset
wal_det_file <- system.file('extdata', 'walleye_detections.csv', package = 'glatos')
```

Next, we will load data from *walleye_detection.csv* using *read_glatos_detections()* and view the structure of the resulting data frame.

```
# Attach glatos package to global environment.
library(glatos)
#> version 0.2.7.9000 ('three-meat-pizza')

# Read in the walleye_detections.csv file using `read_glatos_detections()`
walleye_detections <- read_glatos_detections(wal_det_file)

# View the structure of walleye_detections using `str(walleye_detections)`
str(walleye_detections)
#> Classes 'glatos_detections' and 'data.frame':    7180 obs. of  30 variables:
#>  $ animal_id                : chr  "153" "153" "153" "153" ...
#>  $ detection_timestamp_utc  : POSIXct, format: "2012-04-29 01:48:37" "2012-04-29 01:52:55" ...
#>  $ glatos_array             : chr  "TTB" "TTB" "TTB" "TTB" ...
#>  $ station_no               : chr  "2" "2" "2" "2" ...
#>  $ transmitter_codespace    : chr  "A69-9001" "A69-9001" "A69-9001" "A69-9001" ...
#>  $ transmitter_id           : chr  "32054" "32054" "32054" "32054" ...
#>  $ sensor_value             : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ sensor_unit              : chr  NA NA NA NA ...
#>  $ deploy_lat               : num  43.4 43.4 43.4 43.4 43.4 ...
#>  $ deploy_long              : num  -84 -84 -84 -84 -84 ...
#>  $ receiver_sn              : chr  "113213" "113213" "113213" "113213" ...
#>  $ tag_type                 : chr  NA NA NA NA ...
#>  $ tag_model                : chr  NA NA NA NA ...
#>  $ tag_serial_number        : chr  NA NA NA NA ...
#>  $ common_name_e            : chr  "walleye" "walleye" "walleye" "walleye" ...
#>  $ capture_location         : chr  "Tittabawassee River" "Tittabawassee River" "Tittabawassee River
#>  $ length                   : num  0.565 0.565 0.565 0.565 0.565 0.565 0.565 0.565 0.565 0.565 ...
#>  $ weight                   : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ sex                      : chr  "F" "F" "F" "F" ...
#>  $ release_group            : chr  NA NA NA NA ...
#>  $ release_location         : chr  "Tittabawassee" "Tittabawassee" "Tittabawassee" "Tittabawassee"
```

```
#>  $ release_latitude          : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ release_longitude         : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ utc_release_date_time     : POSIXct, format: "2012-03-20 20:00:00" "2012-03-20 20:00:00" ...
#>  $ glatos_project_transmitter: chr  "HECWL" "HECWL" "HECWL" "HECWL" ...
#>  $ glatos_project_receiver   : chr  "HECWL" "HECWL" "HECWL" "HECWL" ...
#>  $ glatos_tag_recovered      : chr  "NO" "NO" "NO" "NO" ...
#>  $ glatos_caught_date        : Date, format: NA NA ...
#>  $ station                   : chr  "TTB-002" "TTB-002" "TTB-002" "TTB-002" ...
#>  $ min_lag                   : num  258 137 90 90 115 145 106 106 110 110 ...
```

The result is an object with 30 columns (including the columns described above) and two classes: *glatos_detections* and *data.frame*. The *glatos_detections* class label indicates that the data set was created using a glatos load function and therefore should work with any *glatos* function that accepts detection data as input.

Detections obtained from the Ocean Tracking Network

The *read_otn_detections()* function reads in detection data (*.csv files) obtained from the Ocean Tracking Network and reformats the data to meet requirements of *glatos* functions. Data are read using *fread* in the *data.table* package, timestamps are formatted as class *POSIXct* and dates are formatted as class *Date*.

```
# Read in the blue_shark_detections.csv file using `read_glatos_detections()`
shrk_det_file <- system.file("extdata", "blue_shark_detections.csv",
                         package = "glatos")

# Read in the blue_shark_detections.csv file using `read_otn_detections()`
blue_shark_detections <- read_otn_detections(shrk_det_file)


# View the structure of blue_shark_detections using `str(blue_shark_detections)`.
str(blue_shark_detections)
#> Classes 'glatos_detections' and 'data.frame':    3000 obs. of  34 variables:
#>  $ collectioncode         : chr  "NSBS" "NSBS" "NSBS" "NSBS" ...
#>  $ animal_id              : chr  "NSBS-Hooker" "NSBS-Hooker" "NSBS-Hooker" "NSBS-Hooker" ...
#>  $ scientificname         : chr  "Prionace glauca" "Prionace glauca" "Prionace glauca" "Prionace gla
#>  $ commonname             : chr  "blue shark" "blue shark" "blue shark" "blue shark" ...
#>  $ datelastmodified       : Date, format: "2014-12-18" "2014-12-18" ...
#>  $ detectedby             : chr  "HFX" "HFX" "HFX" "HFX" ...
#>  $ receiver_sn            : chr  "HFX" "HFX" "HFX" "HFX" ...
#>  $ glatos_array           : chr  "HFX047" "HFX047" "HFX047" "HFX047" ...
#>  $ receiver               : chr  "146" "146" "146" "146" ...
#>  $ bottom_depth           : num  151 151 151 151 151 151 151 151 151 151 ...
#>  $ receiver_depth         : num  146 146 146 146 146 146 146 146 146 146 ...
#>  $ transmitter_id         : chr  "A69-9001-24395" "A69-9001-24395" "A69-9001-24395" "A69-9001-24395"
#>  $ transmitter_codespace  : chr  "A69-9001" "A69-9001" "A69-9001" "A69-9001" ...
#>  $ sensorname             : chr  NA NA NA NA ...
#>  $ sensorraw              : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ sensortype             : chr  "pinger" "pinger" "pinger" "pinger" ...
#>  $ sensorvalue            : num  NA NA NA NA NA NA NA NA NA NA ...
#>  $ sensorunit             : chr  NA NA NA NA ...
#>  $ detection_timestamp_utc: POSIXct, format: "2014-08-29 06:11:09" "2014-08-29 06:14:43" ...
#>  $ timezone               : chr  "UTC" "UTC" "UTC" "UTC" ...
```

```
#>  $ deploy_long          : num  -63.2 -63.2 -63.2 -63.2 -63.2 ...
#>  $ deploy_lat           : num  44.2 44.2 44.2 44.2 44.2 ...
#>  $ st_setsrid_4326      : chr  "0101000020E61000008A1F63EE5A9E4FC04F1E166A4D1B4640" "0101000020E61
#>  $ yearcollected        : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
#>  $ monthcollected       : int  8 8 8 8 8 8 8 8 8 8 ...
#>  $ daycollected         : int  29 29 29 29 29 29 29 29 29 29 ...
#>  $ julianday            : int  241 241 241 241 241 241 241 241 241 241 ...
#>  $ timeofday            : num  6.19 6.25 6.27 6.3 6.33 ...
#>  $ datereleasedtagger   : Date, format: NA NA ...
#>  $ datereleasedpublic   : Date, format: NA NA ...
#>  $ local_area           : chr  "HALIFAX" "HALIFAX" "HALIFAX" "HALIFAX" ...
#>  $ notes                : chr  NA NA NA NA ...
#>  $ citation             : chr  "Hebert, D., Barthelotte, J., O'Dor, R., Stokesbury, M., Branton, R
#>  $ unqdetecid           : chr  "HFX-A69-9001-24395-180148" "HFX-A69-9001-24395-180149" "HFX-A69-90
```

The result shares the same classes as the walleye dataset (*glatos_detections* and *data.frame*), but has 34 columns, most of which are not shared between OTN and GLATOS data sets. Thus, *read_otn_detections* has only altered those OTN-specific columns needed to meet requirements of *glatos* functions.

Detections exported in CSV format from VEMCO VUE software

There is currently no *glatos* function to load detection data from VEMCO VUE software, so data in that format will need to be loaded using other functions and then carefully checked that it meets requirements described above.

In the example below, we will use the base R functions *read.csv()* and *as.POSIXct()* to load detection data from a csv file and reformat the data to
be consistent with the schema described above. We will then re-load the data using *fread* in the package *data.table* to show advantages of that function versus *read.csv()*. Similarly, we will show advantages of *fast_strptime()* in the *lubridate* package for coercing timestamps from *character* to *POSIXct*.

The first step will be to get the path to a file (*.csv) that contains detection data exported from VEMCO VUE software. Such a file is not included in the *glatos* package. However, we can create such a file using the *glatos* function *vrl2csv()* which exports detection data from a VEMCO *.vrl file using VUE's built-in command line conversion program.

```
#get path to example VRL in this package
vrl_file <- system.file("extdata", "VR2W_109924_20110718_1.vrl",
                        package = "glatos")

#convert the vrl file to csv
#note that vrl2csv writes the csv to disk and returns the path to the csv
csv_file <- vrl2csv(vrl_file) #file name input
```

Now that we have the path to a VUE export file, we will read the data using *read.csv()* and we will measure the elapsed time using the base R function *system.time()*. In this case we are also setting some *read.csv()* arguments to non-default values. First, we set `as.is = TRUE` so that character values are treated as characters and not converted to factors. Second, we set `check.names = FALSE` to prevent conversion of syntactically-invalid column names to syntactically valid names. This simply keeps the names exactly as they appear in the source text file rather than, for example, replacing spaces with ".". This does mean that we need to wrap those column names in backticks when called (e.g., ``my_det$`Sensor Value` ``). Third, we set `fileEncoding = "UTF-8-BOM"` to match the encoding of the text file. If this argument is omitted then you might see the special characters `ï»¿` added to the first column name. Setting the *fileEncoding* may also slow down the import.

```r
t0 <- system.time(
        dtc <- read.csv(csv_file, as.is = TRUE, check.names = FALSE,
                        fileEncoding = "UTF-8-BOM")
      )
t0
#>    user  system elapsed 
#>     0.3     0.0     0.3
```

The operation took 0.3 seconds. Let's look at the structure.

```r
str(dtc)
#> 'data.frame':    69708 obs. of  10 variables:
#>  $ Date and Time (UTC): chr  "2011-04-11 20:17:49" "2011-05-08 05:38:32" "2011-05-08 05:41:09" "2011
#>  $ Receiver           : chr  "VR2W-109924" "VR2W-109924" "VR2W-109924" "VR2W-109924" ...
#>  $ Transmitter        : chr  "A69-1303-63366" "A69-9002-4043" "A69-9002-4043" "A69-9002-4043" ...
#>  $ Transmitter Name   : logi  NA NA NA NA NA NA ...
#>  $ Transmitter Serial : logi  NA NA NA NA NA NA ...
#>  $ Sensor Value       : int  NA 5 7 4 5 16 5 6 4 4 ...
#>  $ Sensor Unit        : chr  "" "ADC" "ADC" "ADC" ...
#>  $ Station Name       : logi  NA NA NA NA NA NA ...
#>  $ Latitude           : int  0 0 0 0 0 0 0 0 0 0 ...
#>  $ Longitude          : int  0 0 0 0 0 0 0 0 0 0 ...
```

Now we we will reformat to be consistent with a *glatos_detections* object.

```r
#change column name "Date and Time (UTC)" to "detection_timestamp_utc"
names(dtc)[1] <- "detection_timestamp_utc"

#change column name "Receiver" to "receiver_sn"
names(dtc)[2] <- "receiver_sn"
```

```r
str(dtc)
#> 'data.frame':    69708 obs. of  10 variables:
#>  $ detection_timestamp_utc: chr  "2011-04-11 20:17:49" "2011-05-08 05:38:32" "2011-05-08 05:41:09" "
#>  $ receiver_sn            : chr  "VR2W-109924" "VR2W-109924" "VR2W-109924" "VR2W-109924" ...
#>  $ Transmitter            : chr  "A69-1303-63366" "A69-9002-4043" "A69-9002-4043" "A69-9002-4043" ..
#>  $ Transmitter Name       : logi  NA NA NA NA NA NA ...
#>  $ Transmitter Serial     : logi  NA NA NA NA NA NA ...
#>  $ Sensor Value           : int  NA 5 7 4 5 16 5 6 4 4 ...
#>  $ Sensor Unit            : chr  "" "ADC" "ADC" "ADC" ...
#>  $ Station Name           : logi  NA NA NA NA NA NA ...
#>  $ Latitude               : int  0 0 0 0 0 0 0 0 0 0 ...
#>  $ Longitude              : int  0 0 0 0 0 0 0 0 0 0 ...
```