Alaska
Fisheries Science
Center

National Marine
Fisheries Service

U.S DEPARTMENT OF COMMERCE

AFSC PROCESSED REPORT 2017-01
doi:10.7289/V5/AFSC-PR-2017-01

Primer for Archival Tag Data Processing Using R

January 2017

This document should be cited as follows:

Coutré, K. M. 2017. Primer for archival tag data processing using R. AFSC Processed Rep. 2017-01, 21 p. Alaska Fish. Sci. Cent., NOAA, Natl. Mar. Fish. Serv., 7600 Sand Point Way NE, Seattle WA 98115.

Available at http://www.afsc.noaa.gov/Publications/ProcRpt/PR2017-01.pdf

Primer for Archival Tag Data Processing Using R

K. M. Coutré

Under contract to Auke Bay Laboratories
Alaska Fisheries Science Center
National Marine Fisheries Service
National Oceanic and Atmospheric Administration
17109 Point Lena Loop Road
Juneau, AK, 99801

January 2017

ABSTRACT

This report provides detailed steps for downloaded archival tag data processing in Microsoft Excel and R. Depth and temperature time series data from implanted archival tags in Greenland turbot (*Reinhardtius hippoglossoides*) are used as a case study. Steps outlined include formatting measurements and standardizing units and time zones. After initial error checking in Microsoft Excel, the procedures for importing, compiling, formatting, and saving data in R are described. Instructions and R code are provided to filter the data and calculate basic metrics. Code is provided to create and save data visualizations that aid in the understanding of the data, detecting patterns, and identifying errors.

CONTENTS

ABSTRACT iii

BACKGROUND 2

DATA ERROR AND CHECKING AND PROCESSING 2

Step 1. Format the Raw Measurements 2

Step 2. Unify Structure Across Tags 3

Step 3. Error Check the Tag Data 4

Step 4. Import Tag Data into R 4

Step 5. Format Data in R 5

Step 6. Import and Format Release and Recapture Data 6

Step 7. Subset Data to Relevant Detections 7

Step 8. Calculate Basic Metrics 8

Step 9. Save the Filtered Data Frame 9

Step 10. Exploratory Plotting 10

Step 11. Save Plots 14

ACKNOWLEDGMENTS 15

CITATIONS 23

APPENDIX 19

BACKGROUND

Archival tags, also called data storage tags (DST), are electronic tags that are internally or externally attached to animals to record data such as temperature, depth, heart rate, acceleration, light level, and movement rate at assigned time intervals. In fisheries research, archival tags have been used to gain insight into fish behavior such as fine-scale vertical movement, changes in fish depth as related to environmental conditions such as tide and daylight cycles, and large-scale vertical and seasonal migrations (Horodysky et al. 2007, Le Port et al. 2008, Schaefer and Fuller 2010, Boje et al. 2014). Stock assessment scientists have also used archival tag data to inform the availability of a species to commercial fishing and/or survey gear (Nichol and Somerton 2002, Sippel et al. 2015, Hulson et al. 2016). Technological advances of archival tag data storage capabilities and reduced tag sizes are making archival tags a viable option for gathering movement information on an increasing variety of fish species.

The Marine Ecology and Stock Assessment (MESA) program at Auke Bay Laboratories (ABL) in Juneau, Alaska, administers the groundfish tag program for the Alaska Fisheries Science Center (AFSC). MESA maintains a tag database which includes information from fish tagged with conventional tags, pop-up satellite archival tags, and internally implanted archival tags. Since the 1990s, MESA has released a total of 1,729 sablefish *Anoplopoma fimbria*, 297 Greenland turbot *Reinhardtius hippoglossoides*, 203 shortspine thornyhead *Sebastolobus alascanus*, 170 lingcod *Ophiodon elongates* and 178 Pacific spiny dogfish *Squalus suckleyi* with archival tags in the Gulf of Alaska and Bering Sea. Because archival tags can provide a large quantity of raw data, simplified methods for preparing the data for analyses are needed. The objective of this report is to provide step-by-step procedures for processing archival data into a format ready for quantitative analyses and for preliminary data exploration. Depth and

temperature data, recovered from nine archival tags (Lotek Wireless Inc.) implanted in Greenland turbot, are used as a case study. The turbot were tagged in the Bering Sea during the AFSC Longline Survey (Rutecki et al. 2016). Tags recorded depth and temperature every 1 minute or 15 minutes during the fish's time at liberty. The turbot example data presented in this report was obtained from archival tags; however, many of the steps presented here could be applied to time series datasets of other origins as well. Computer code created in the software environment R (R Core Team 2015) is presented and serves as an example for data setup and exploration, which are methodological steps generally considered too detailed for scientific journal publications. The coding examples also provide exposure to various useful R packages for data formatting and manipulation. Basic proficiency using R is needed to implement the methods described within this report and can be achieved using the following manual: An Introduction to R (Venables et al. 2016; http://cran.us.r-project.org/doc/manuals/R-intro.pdf).

DATA ERROR AND CHECKING AND PROCESSING

Step 1. Format the Raw Measurements

When working with data from multiple tags and potentially multiple tag companies, one must first standardize all data and ensure uniform formatting. All measurement units should be verified and conversion calculations performed for differing measurement units. In our case, some tags reported pressure in pounds per square inch (PSI), and others in decibars (dbars). In order to convert pressure to meters of seawater (depth), readings in PSI were multiplied by 0.69; readings in dbars were kept as is because 1 dbar converts to ~ 1 m seawater. Measurement unit conversions are readily available on the internet from various sources including: http://docs.bluerobotics.com/calc/pressure-depth/. Time and date values must be standardized so

that tag data can be synched among tags and with other types of data such as tidal cycle and daylight. With the turbot data, time was standardized by converting to Coordinated Universal Time or UTC.

Step 2. Unify Structure Across Tags

To prepare data for use in the software program R, a CSV (comma-separated value) file of each tag's data should be made with appropriate column headings for each data field. In our turbot example, we used: TAG_NUM (a unique tag identifying number), Time_UTC (date and time in UTC), Temp_C (temperature in degrees Celsius), and Depth_M (depth in meters). An example is shown below with the first 15 rows of file "GG390.csv": Note that depth is a negative value in the example because those initial detections are slightly above the surface while the fish is being tagged. These pre-release values will be filtered out later.

	A	B	C	D
1	TAG_NUM	Time_UTC	Temp_C	Depth_M
2	390	5/17/2011 0:00	20.78	-1.25
3	390	5/17/2011 0:01	20.76	-1.25
4	390	5/17/2011 0:02	20.76	-0.75
5	390	5/17/2011 0:03	20.76	-1.25
6	390	5/17/2011 0:04	20.76	-1.25
7	390	5/17/2011 0:05	20.74	-0.75
8	390	5/17/2011 0:06	20.74	-0.75
9	390	5/17/2011 0:07	20.72	-0.75
10	390	5/17/2011 0:08	20.72	-0.75
11	390	5/17/2011 0:09	20.72	-0.75
12	390	5/17/2011 0:10	20.72	-1.25
13	390	5/17/2011 0:11	20.7	-0.75
14	390	5/17/2011 0:12	20.7	-0.75
15	390	5/17/2011 0:13	20.7	-1.25

Step 3. Error Check the Tag Data

Once formatting has occurred, tag depth and temperature data should be manually compared with release and recapture data to verify that the detection information from the tag matches the time the tag was at liberty. Time intervals between data records should also be checked to make sure they are logical. For example, archival tags often record at regular intervals on the scale of seconds to hours.

Step 4. Import Tag Data into R

To process the archival tag data, create a new R script is created and set the working directory to the location where all pertinent files are located using the “setwd” function followed by the full file path to the working directory. Example R code is presented below. The names of the CSV data files are saved as a list called “files” and then each one is imported (“read.csv” function) and combined into a data frame called “tag_data” (using “rbind”). The “do.call” function facilitates using the “rbind” function on the list of files being imported. All of the column headings in the CSV files must be identical for the below method.

```
setwd( "\\\\nmfs.local/AKC-ABL/Users2/karson.coutre/Desktop/R_Analysis"
)
files <- c("GG386.csv", "GG387.csv", "GG390.csv","GG403.csv","GG4122.cs
v","GG4126.csv","GG5364.csv","GG5319.csv","GG5282.csv")

tag_data <- do.call("rbind",lapply(files,
                                   FUN=function(files){read.csv(files,
                                                                header=T
RUE, sep=",")}))
```

For more information on the different functions used within this report, typing a “?” followed by the name of any function in R will open a help file that includes information such as the arguments within the parentheses and the expected output for that function. For example,

```
?setwd

?do.call
```

.

Step 5. Format Data in R

To designate a variable as a factor for use in data analyses, use the “as.factor” command. In the example below, the data field “TAG_NUM” in the data frame “tag_data” is set as non-numeric so that during plotting and analysis R will treat this column as a category such as fish1, fish2, and fish3 rather than a numerical gradient.

```
Tag_data$TAG_NUM <- as.factor(tag_data$TAG_NUM)
```

.

In our case, the “Depth_M” data field had commas inserted in numbers greater than 1,000. In order for any column in a data frame to be classified as numeric, commas must be removed from all data within that column. To accomplish this, the “gsub” function for replacement can be used to remove commas from the “Depth_M” column. In the example below “gsub” substitutes all commas from the designated column with nothing, which deletes the comma. Once the commas are removed, the values can be designated as numeric,

```
tag_data$Depth_M <- as.numeric(gsub(",","",tag_data$Depth_M))
```

.

Format the time using the package “lubridate” (Grolemund and Wickham 2011). This package has many functions that work with dates and times to streamline formatting and perform various calculations. The “Time_UTC” column that was imported into R from the CSV files must first be converted to a character string to format the data values as date time object using

the function “mdy_hm”. Within the function “mdy_hm” the time zone is also designated (argument “tz”),

```
Library(lubridate)
tag_data$Time_UTC <- as.character(tag_data$Time_UTC)
tag_data$Time_UTC <- mdy_hm(tag_data$Time_UTC, tz=”UTC”)
```

.

Step 6. Import and Format Release and Recapture Data

After the basic formatting of the “tag_data” dataframe, the release and recapture data are imported and formatted similarly. The following is an image of the release_recap.csv file imported into R with “PRIM_TAG_NUM” as the tag number (same as “TAG_NUM” in the tag_data data frame), “HAUL_DATE” as the release date of the tagged fish, and “REC_DATE” as the recovery date of the tagged fish.

	A	B	C
1	PRIM_TAG_NUM	HAUL_DATE	REC_DATE
2	386	5/30/2011 0:00	7/3/2012 0:00
3	387	5/30/2011 0:00	7/4/2011 0:00
4	390	5/30/2011 0:00	7/25/2011 0:00
5	403	5/30/2011 0:00	7/13/2011 0:00
6	406	5/30/2011 0:00	8/18/2014 0:00
7	4122	6/10/2003 0:00	7/8/2004 0:00
8	4126	6/15/2003 0:00	6/3/2004 0:00
9	5282	6/4/2003 0:00	6/9/2006 0:00
10	5319	6/6/2003 0:00	7/7/2004 0:00

In the R code below, the “read.csv” function is used to import the file “release_recap.csv” and save it as a data frame called “RR.” In the second and third lines, the fields “HAUL_DATE” and “REC_DATE” are converted to character strings and then date times using the same formatting of the time column (“TIME_UTC”) in step 5,

```
RR <- read.csv("release_recap.csv")
RR$HAUL_DATE <- mdy_hm(as.character(RR$HAUL_DATE), tz="UTC")
RR$REC_DATE <- mdy_hm(as.character(RR$REC_DATE), tz="UTC")
```

.

Step 7. Subset Data to Relevant Detections

Tag data recorded prior to release and post-recovery should be removed from the dataset. To accomplish this step, a “for” loop is used which automates a repetitive process. The goal of the loop is to extract the release and recapture date for each tag number (using dataframe “RR”) and use those dates to subsequently extract the data from the data frame “tag_data” only for the period when the fish was at liberty. This subset of the data is saved as a separate dataframe.

The first line of code below creates an empty data frame which will be filled by output from the loop. The second line of code initiates the “for” loop and designates how many times to iterate through the code within the curly brackets, so in this case the loop will repeat the steps for each row of the “RR” data frame.

All of the code inside the curly braces (e.g., “{“) will be looped. The first line in the loop creates an object called “tag_num” which represents a different tag number (“PRIM_TAG_NUM”) for each iteration of the loop. The second and third lines extract the release date and recapture date and saves them as “start” and “end” for the assigned tag number. The next lines extract all of the detections for a specific tag where the time column is greater than or equal to the start time and less than or equal to the end time and saves them in an output data frame (f_tag_data). Using the “rbind” function each iteration of the loop adds each tag number’s subset of detections to the output data frame,

```
F_tag_data <- data.frame()

for (I in 1:nrow(RR))
```

```
{
  tag_num <- RR[I,"PRIM_TAG_NUM"]
  start <- RR[RR$PRIM_TAG_NUM == tag_num,"HAUL_DATE"]
  end <- RR[RR$PRIM_TAG_NUM == tag_num,"REC_DATE"]


  f_tag_data <- rbind(f_tag_data,
                 tag_data[tag_data$TAG_NUM == tag_num &
                           tag_data$Time_UTC >= start &
                           tag_data$Time_UTC <= end,])


}
```

.

After filtering the relevant data, or at multiple steps of data import and formatting, check for NAs in any of the columns using the "is.na" function. In the code below, "Depth_M" can be replaced with any of the column names to look for NAs. The "str" function identifies how many times and where those NAs occur,

```
na_df <- tag_data[is.na(f_tag_data$Depth_M),]
str(na_df)
```

.

Step 8. Calculate Basic Metrics

It is also important to check some basic calculations such as the mean and the range while processing the data to make sure there are no glaring errors or implausible values. The "summary" function calculates the minimum, maximum, 1st and 3rd quartile, median and mean when given a vector of numbers. The output of the mean and range functions included below will be NA if there are any NAs within the column being calculated. This can be remedied by using the function "na.omit",

```
range(f_tag_data$Depth_M)
mean(f_tag_data$Depth_M)
```

```
summary(f_tag_data$Depth_M)

na.omit(mean(f_tag_data$Depth_M))
```

.

The mean depth and other metrics can also be calculated by individual fish using the function “ddply” in the “plyr” package (Wickham 2011) which is useful for data manipulation. In the example code below, “ddply” is used to create a data frame (mean_depth) with one row for each tag (TAG_NUM) and the corresponding mean and standard deviation of the “Depth_M” column. The output data frame is shown below the code,

```
library(plyr)
mean_depth <- ddply(f_tag_data, .(TAG_NUM), summarize, ind_mean=mean(De
pth_M), sdev=sd(Depth_M))
```

.

Data frame “mean_depth”:

```
   TAG_NUM ind_mean      sdev
      386  450.5429  203.84635
      387  531.5494  115.05434
      390  457.5057   84.41713
      403  563.1171  119.06803
     4122  566.6892  131.71609
     4126  576.0446  174.39686
     5282  490.3147  286.86321
     5319  750.9997  276.33416
     5364  703.2208  281.05200
```

.

Step 9. Save the Filtered Data Frame

The save function will save the two data frames that have been formatted above to the working directory as R datasets. This makes it easy to share with others or start analysis at a later R session. The load function can then be used at the beginning of an analysis script file or at the start of a new R session,

```
Save(RR, file="release_recapture.Rda")
save(f_tag_data, file="archival_turbot.Rda")

load("release_recapture.Rda")
load("archival_turbot.Rda")
```

.

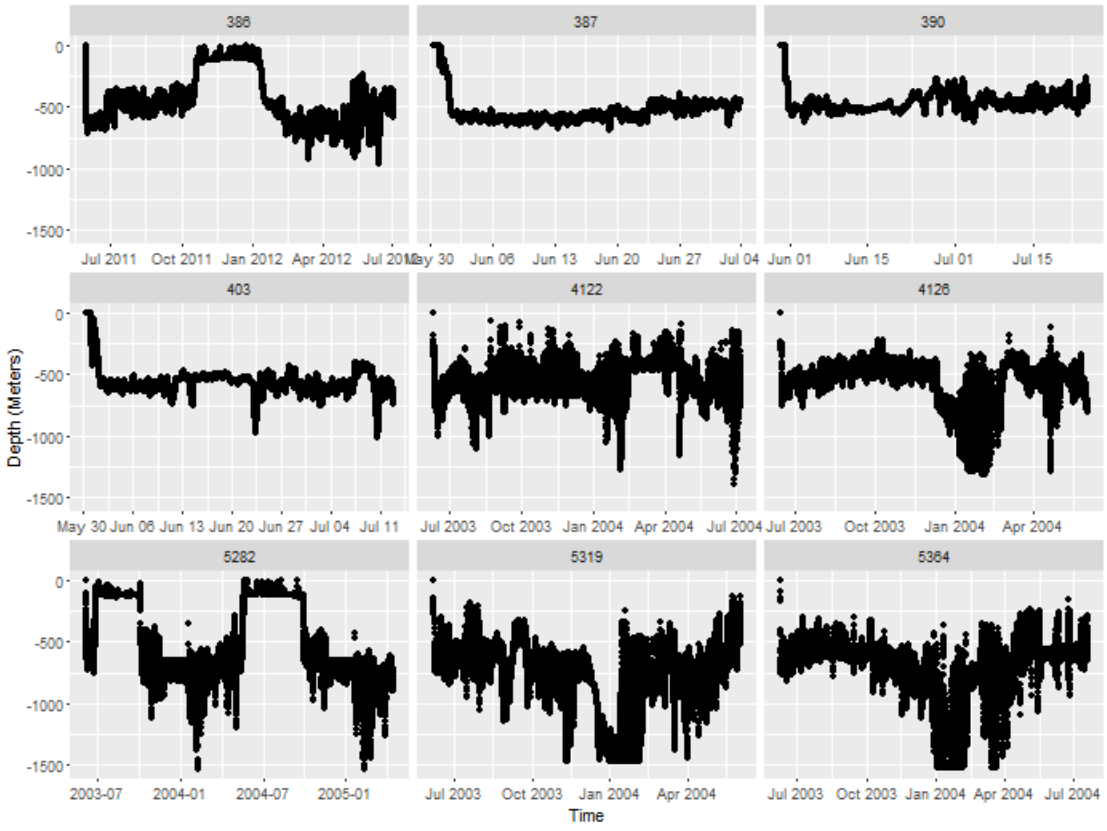
Step 10. Exploratory Plotting

This section provides data visualizations using the processed data frame "f_tag_data" which was saved as "archival_turbot.Rda" in the working directory. Although there are many tools to visualize data in R, these examples use the package "ggplot2" (Wickham 2009) for plotting. To create a plot using "ggplot2" notation, begin with the function "ggplot" to initiate a plot and provide the name of the data frame containing data you want to plot. Next, within the "ggplot" function use the "aes" function to provide the column names for the x- and y-axes, respectively. Other features of the plot are added using a "+" outside of the ggplot function (Wickham 2009). The most important feature to add is the "geom_xxxx" which selects the type of plot, such as boxplot ("geom_boxplot"), points ("geom_point"), or lines ("geom_line"). Other features include adding labels using the function "labs" or jittering points on a plot with "position_jitter" (Wickham 2009). This notation becomes clearer after a few examples and more information on customizing plots can be found here: http://docs.ggplot2.org/current/. The exploratory plots within this step will aid in understanding a dataset and detecting patterns or potential errors.

To plot depth over time for individual tags, the facet_wrap function from the "ggplot2" package is used to create multiple panels within one graphic. Here the panels represent the unique "TAG_NUM" in order to look at depth by fish. In each panel, the x-axis (time) has differing scales depending on the time at liberty for each fish. This is set by the argument "scales

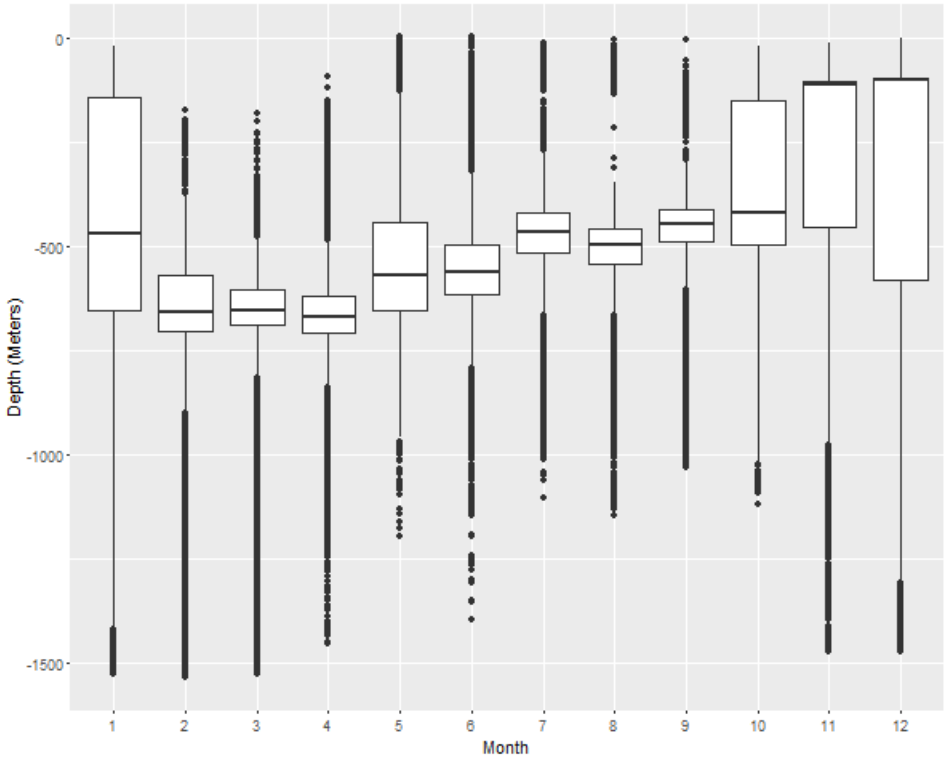
= “free x”.” A negative sign is entered before the depth column (“Depth_M”) when assigning the y-axis for intuitive depth visualization with zero being the surface of the water. By default the y-axis is fixed so depth ranges are comparable among panels,

```
ggplot(f_tag_data, aes(Time_UTC, -Depth_M)) +
  geom_point() +
  facet_wrap(~TAG_NUM, scales= "free_x") +
  labs(x="Time", y="Depth (Meters)")
```



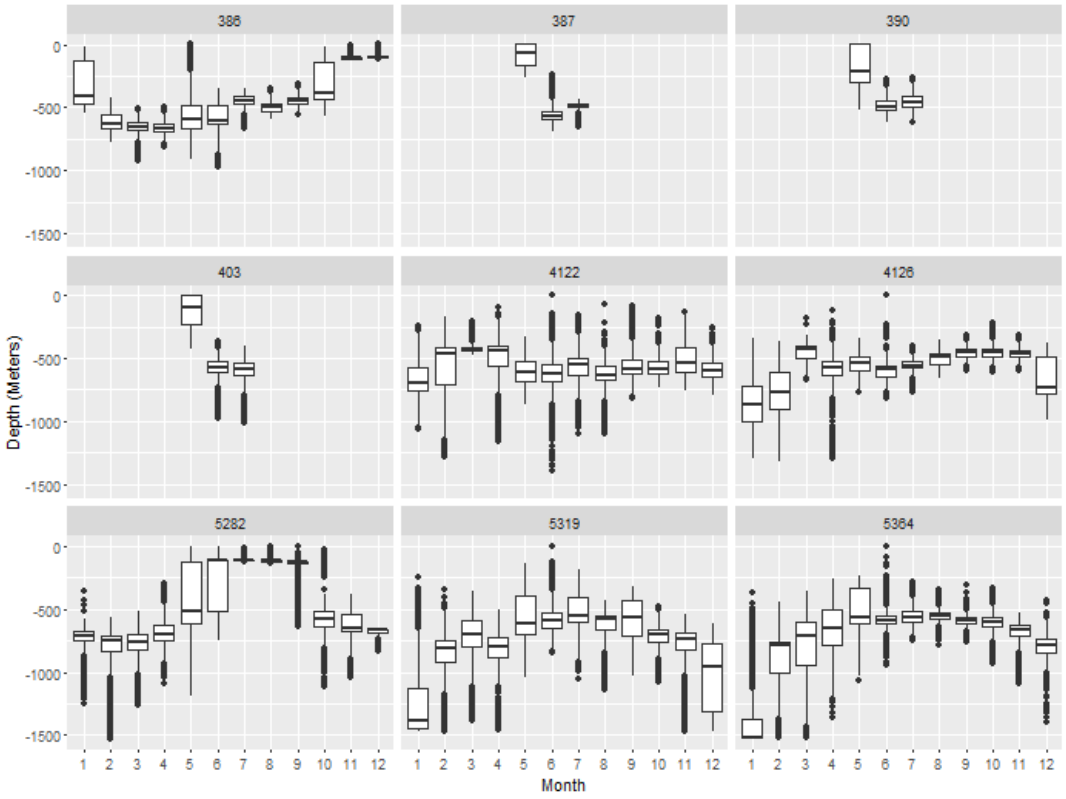
In order to visualize seasonal patterns in depth distributions among fish throughout a calendar year, create a monthly boxplot of depth with tags combined. First the “month” function is used to extract the month from the time column, then a column is added to the detection data frame called “Month” to be used as the x-axis for plotting,

```
library(ggplot2)
library(lubridate)

f_tag_data$Month<-month(f_tag_data$Time_UTC)


ggplot(f_tag_data, aes(factor(Month),-Depth_M))+
  geom_boxplot() +
  labs(x="Month", y="Depth (Meters)")
```



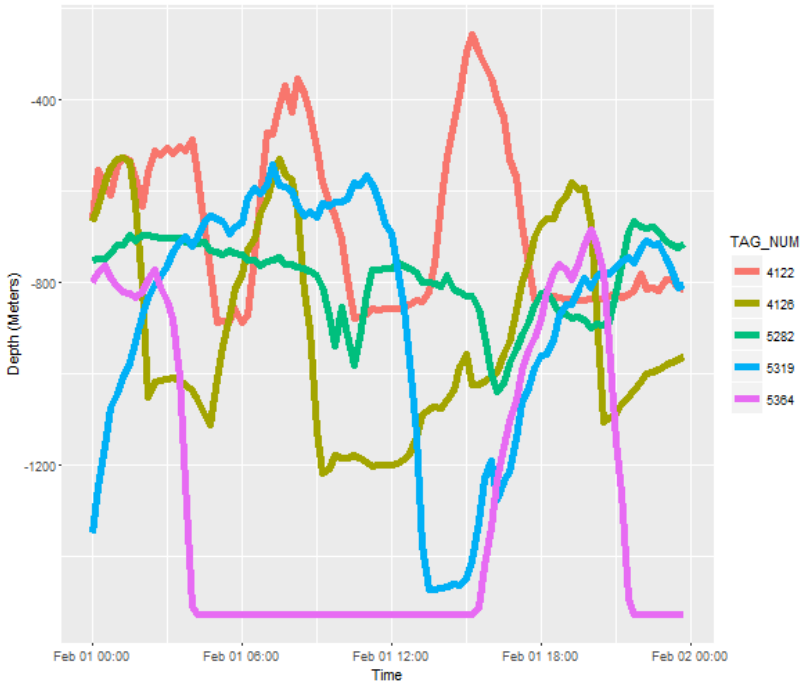
Using the “facet_wrap” function again, monthly boxplots for each individual fish can be more informative if there is a lot of individual variation.

```
ggplot(f_tag_data, aes(factor(Month), -Depth_M)) +
  geom_boxplot() +
  facet_wrap(~TAG_NUM) +
  labs(x="Month", y="Depth (Meters)")
```



To qualitatively assess synchrony in depth movements across fish, the data can be filtered by a specific time interval and viewed for all tags on the same plot. This example subsets one day, 1 February, 2004; however, any time interval can be inserted into this code. The function "as.POSIXct" allows the text in quotations to represent a date and time in order to subset the time interval. In the R code for the plot using "ggplot," the "color" argument assigns which variable the colored lines will represent, while "size" designates the size of the lines on the plot,

```
feb1 <- f_tag_data[f_tag_data$Time_UTC >= as.POSIXct('2004-02-01 00:00:
00', tz="UTC") & f_tag_data$Time_UTC < as.POSIXct('2004-02-02 00:00:00'
, tz="UTC"),]


ggplot(feb1, aes(Time_UTC, -Depth_M, color=TAG_NUM)) +
  geom_line(size=1.8) +
  labs(x="Time", y="Depth (Meters)")
```



Step 11. Save Plots

Lastly, the function “ggsave” allows one to easily save figures created with “ggplot2” in a variety of file types including .pdf, .jpeg, .tiff, and .png. This function defaults to the most recently plotted graphic and saves the file to the working directory. This function allows graphics to be highly customizable which is useful for manuscript figures. Here are two examples:

```
ggsave(file="Feb_1_plot.pdf", width=5, height=4)
ggsave(file="Feb_1_plot.tiff", dpi=300)
```

.

ACKNOWLEDGMENTS

I thank Katy Echave for her useful feedback and information from the sablefish tag database, Pat Malecha and Jon Heifetz for insightful editing, and Jim Ianelli for providing the turbot tag data.

CITATIONS

Boje, J., S. Neuenfeldt, C. R. Sparrevohn, O. Eigaard, and J. W. Behrens. 2014. Seasonal migration, vertical activity, and winter temperature experience of Greenland halibut *Reinhardtius hippoglossoides* in West Greenland waters. Mar. Ecol. Progr. Ser. 508:211-222.

Grolemund, G., and H. Wickham. 2011. Dates and times made easy with lubridate. J. Stat. Softw. 40(3): 1-25. URL http://www.jstatsoft.org/v40/i03/.

Horodysky, A. Z., D. W. Kerstetter, R. J. Latour, and J. E. Graves. 2007. Habitat utilization and vertical movements of white marlin (*Tetrapturus albidus*) released from commercial and recreational fishing gears in the western North Atlantic Ocean: Inferences from short duration pop-up archival satellite tags. Fish. Oceanogr. 16: 240-256.

Hulson, P. J. F., C. A. Tribuzio, and K. M. Coutré. 2016. The use of satellite tags to inform the stock assessment of a data-poor species: Estimating vertical availability of spiny dogfish in the Gulf of Alaska.. *In* T.J. Quinn II, J.L. Armstrong, M. Baker, J. Heifetz, and D. Witherell (eds.), Assessing and Managing Data-Limited Fish Stocks. Alaska Sea Grant College Program AK-SG-16-01. University of Alaska Fairbanks.

Le Port, A., T. Sippel, and J. C. Montgomery. 2008. Observations of mesoscale movements in the short-tailed stingray, *Dasyatis brevicaudata* from New Zealand using a novel PSAT tag attachment method. J. Exp. Mar. Biol. Ecol. 359: 110-117.

Nichol, D. G., and D. A. Somerton. 2002. Diurnal vertical migration of the Atka mackerel *Pleurogrammus monopterygius* as shown by archival tags. Mar. Ecol. Prog. Ser. 239: 193-207.

R Core Team. 2015. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing.

Rutecki, T. L., C. J. Rodgveller, and C. R. Lunsford. 2016. National Marine Fisheries Service longline survey data report and survey history, 1990-2014. U.S. Dep. Commer., NOAA Tech. Memo. NMFS-AFSC324, 329 p.

Schaefer, K. M., and D. W. Fuller. 2010. Vertical movements, behavior, and habitat of bigeye tuna (*Thunnus obesus*) in the equatorial eastern Pacific Ocean, ascertained from archival tag data. Mar. Biol. 157: 2625-2642.

Sippel, T., T. P. Eveson, B. Galuardi, C. Lam, S. Hoyle, M. Maunder, P. Kleiber, F. Carvalho, V. Tsontos, and S. L. Teo. 2015. Using movement data from electronic tags in fisheries stock assessment: a review of models, technology and experimental design. Fish. Res. 163: 152-160.

Venables, W. N., D. M. Smith, R Development Core Team. 2016. An Introduction to R: Notes on R, A Programming Environment for Data Analysis and Graphics. Version 3.3.1. URL http://cran.r-project.org/doc/manuals/R-intro.pdf.

Wickham, H. 2009. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. URL http://had.co.nz/ggplot2/book.

Wickham, H. 2011. The Split-Apply-Combine Strategy for Data Analysis. J. Stat. Softw. 40(1), 1-29.URL http://www.jstatsoft.org/v40/i01/.

APPENDIX

Below is all of the R code contained within the report with explanatory text removed.

```
setwd( "\\\\nmfs.local/AKC-ABL/Users2/karson.coutre/Desktop/R_Analysis")
files <- c("GG386.csv", "GG387.csv", "GG390.csv","GG403.csv","GG4122.csv","GG
4126.csv","GG5364.csv","GG5319.csv","GG5282.csv")

tag_data <- do.call("rbind",lapply(files,
                                   FUN=function(files){read.csv(files,
                                                                header=TRUE, s
ep=",")}))



?setwd
?do.call


tag_data$TAG_NUM <- as.factor(tag_data$TAG_NUM)
tag_data$Depth_M <- as.numeric(gsub(",","",tag_data$Depth_M))

library(lubridate)
tag_data$Time_UTC <- as.character(tag_data$Time_UTC)
tag_data$Time_UTC <- mdy_hm(tag_data$Time_UTC, tz="UTC")


RR <- read.csv("release_recap.csv")
RR$HAUL_DATE <- mdy_hm(as.character(RR$HAUL_DATE), tz="UTC")
RR$REC_DATE <- mdy_hm(as.character(RR$REC_DATE), tz="UTC")

f_tag_data <- data.frame()

for (i in 1:nrow(RR))
{
  tag_num <- RR[i,"PRIM_TAG_NUM"]
  start <- RR[RR$PRIM_TAG_NUM == tag_num,"HAUL_DATE"]
  end <- RR[RR$PRIM_TAG_NUM == tag_num,"REC_DATE"]


  f_tag_data <- rbind(f_tag_data,
                tag_data[tag_data$TAG_NUM == tag_num &
                           tag_data$Time_UTC >= start &
                           tag_data$Time_UTC <= end,])


}
```

```
na_df <- tag_data[is.na(f_tag_data$Depth_M),]
str(na_df)


range(f_tag_data$Depth_M)
mean(f_tag_data$Depth_M)
summary(f_tag_data$Depth_M)
na.omit(mean(f_tag_data$Depth_M))

library(plyr)
mean_depth <- ddply(f_tag_data, .(TAG_NUM), summarize, ind_mean=mean(Depth_M)
, sdev=sd(Depth_M))

save(RR, file="release_recapture.Rda")
save(f_tag_data, file="archival_turbot.Rda")

load("release_recapture.Rda")
load("archival_turbot.Rda")

ggplot(f_tag_data, aes(Time_UTC, -Depth_M)) +
  geom_point() +
  facet_wrap(~TAG_NUM, scales= "free_x") +
  labs(x="Time", y="Depth (Meters)")


library(ggplot2)
library(lubridate)
f_tag_data$Month<-month(f_tag_data$Time_UTC)

ggplot(f_tag_data, aes(factor(Month),-Depth_M))+
  geom_boxplot() +
  labs(x="Month", y="Depth (Meters)")


ggplot(f_tag_data, aes(factor(Month), -Depth_M)) +
  geom_boxplot() +
  facet_wrap(~TAG_NUM) +
  labs(x="Month", y="Depth (Meters)")

feb1 <- f_tag_data[f_tag_data$Time_UTC >= as.POSIXct('2004-02-01 00:00:00', t
z="UTC") & f_tag_data$Time_UTC < as.POSIXct('2004-02-02 00:00:00', tz="UTC"),
]

ggplot(feb1, aes(Time_UTC, -Depth_M, color=TAG_NUM)) +
  geom_line(size=1.8) +
  labs(x="Time", y="Depth (Meters)")
```

```r
ggsave(file="Feb_1_plot.pdf", width=5, height=4)
ggsave(file="Feb_1_plot.tiff", dpi=300)
```