Creating Interactive Plots in R

Greg Mitchell and Tara L. Crewe

September 12, 2016

This code was written by Greg Mitchell (Environment Canada) and modified into an Rmd file for the Motus website by Tara Crewe (Bird Studies Canada).

Objective and Background

The purpose of this code is to visualize Motus detection data and assess and measure departure times or other temporal parameters. The following function allows the user to zoom in on a plot for data exploration. Once zoomed in to the data of interest, a second function allows the user to measure data from the plot, and store the output in a database. The function defined stores the time stamp of a specified point to file, which can be useful for extracting departure time from a site, for example. The function could be modified for other purposes.

The dataset used in this example is from a single tower on Kent Island, New Brunswick, Canada, and includes all detections from 2010 for a bird with tag ID 155. Data were recorded with a Lotek SRX600 receiver.

```
#setwd("F:/.....") # user may wish to set the working directory
library(lubridate)

tag_155 <- read.csv("TAG_155.csv")

# modify times and sort data by Date_Time
tag_155$Date_Time <- ymd_hms(tag_155$Date_Time, tz="America/New_York")
tag_155$Date_Time <- with_tz(tag_155$Date_Time, tz="UTC")
tag_155 <- tag_155[order(tag_155$Date_Time),]
```

Plot the data

Note that for a given tower, you may wish to look at each antenna separately to get a feel for the direction of movement. In this example there are four antenna, so we split the screen accordingly.

```
n.antenna <- length(unique(tag_155$Antenna))
split.screen(c(n.antenna,1))
```

We then set the parameters for each plot and plot the data. Here, we are using the same parameters for each panel:

1. 'mar' refers to the number of lines of text in the margin around the plot on the bottom, left, top, and right, respectively.

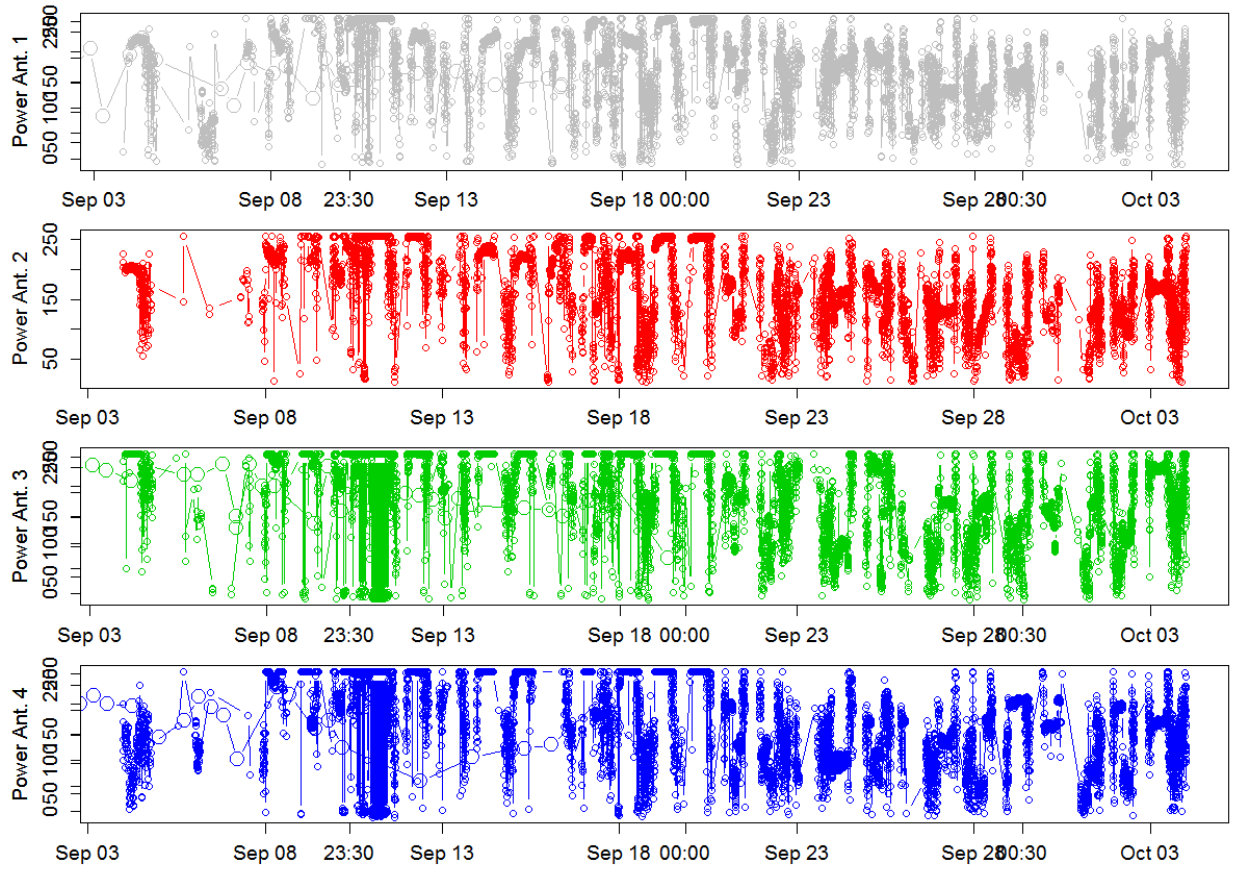
2. 'cex.axis' refers to text size.

3. 'mgp' is the margin line on which axes labels are drawn.

4. 'xpd=T' allows you to plot things like the legend outside of the plot.

5. 'omi' refers to the size of the outer margins of the plots in inches, and allows for finer control relative to mar.

```
screen(1, FALSE)
par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
with(subset(tag_155, Antenna==1), plot(Power ~ Date_Time, type='b', cex=1,
col=8, ylab="Power Ant. 1"))

screen(2, FALSE)
par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
with(subset(tag_155, Antenna==2), plot(Power ~ Date_Time, type='b', cex=1,
col=2, ylab="Power Ant. 2"))

screen(3, FALSE)
par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
with(subset(tag_155, Antenna==3), plot(Power ~ Date_Time, type='b', cex=1,
col=3, ylab="Power Ant. 3"))

screen(4, FALSE)
par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
with(subset(tag_155, Antenna==4), plot(Power ~ Date_Time, type='b', cex=1,
col=4, ylab="Power Ant. 4"))
```



Zooming Function

Now that the data are plotted, we can apply the zooming function. This is a simple function to establish new plotting limits. You will have to adjust the split screen function as necessary.

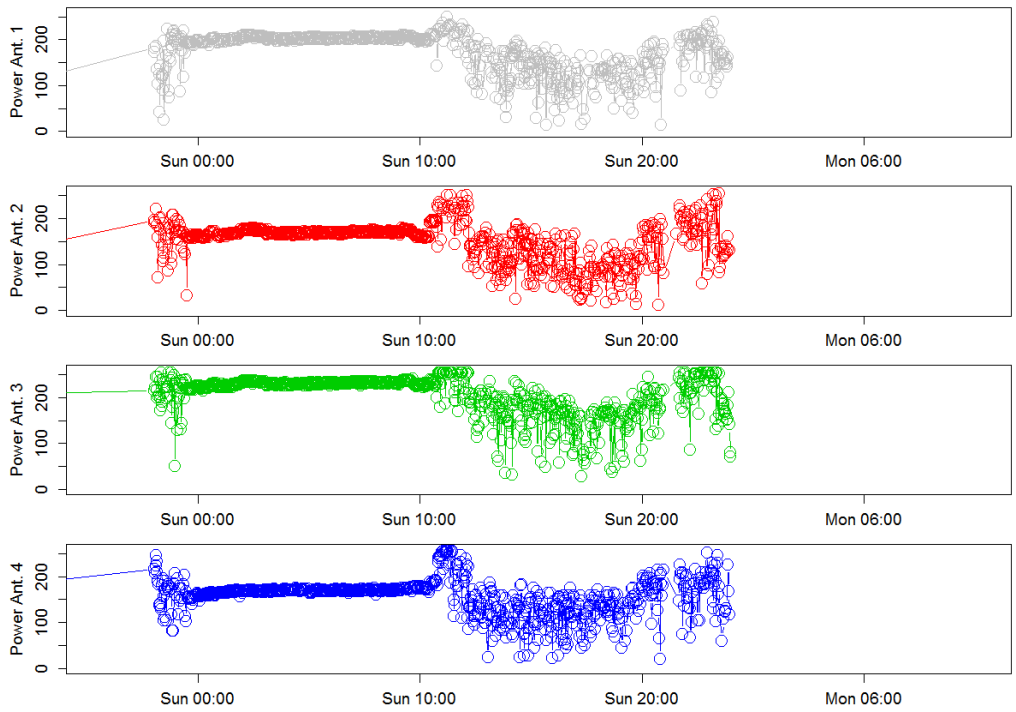
```
zoom = function(x) {  #x = name of dataframe
 reg = locator(4)
 reg$x <- reg$x + origin
 yaxis <- c(max(reg$y)+10, min(reg$y)-10)
 xaxis <- c(max(reg$x), min(reg$x)) #Note - I am not sure why, but taking min
or max of date changes tz.
 xaxis <- with_tz(xaxis, tz="UTC")
 dev.off()

 split.screen(c(n.antenna,1))

 screen(1, FALSE)
 par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
 plot(yaxis~xaxis,type='n', ylab="Power Ant. 1")
```

```
 with(subset(x, Antenna==1), points(Power ~ Date_Time, type='b', cex=2,
col=8))

 screen(2, FALSE)
 par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
 plot(yaxis~xaxis,type='n', ylab="Power Ant. 2")
 with(subset(x, Antenna==2), points(Power ~ Date_Time, type='b', cex=2,
col=2))

 screen(3, FALSE)
 par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
 plot(yaxis~xaxis,type='n', ylab="Power Ant. 3")
 with(subset(x, Antenna==3), points(Power ~ Date_Time, type='b', cex=2,
col=3))

 screen(4, FALSE)
 par(mar=c(2,4,1,1), cex.axis=1.2, cex.lab=1.2, mgp=c(2.5,1,0), xpd=T,
omi=c(0.5,0,0,0), xpd=F)
 plot(yaxis~xaxis,type='n', ylab="Power Ant. 4")
 with(subset(x, Antenna==4), points(Power ~ Date_Time, type='b', cex=2,
col=4))
}
```

Apply Zoom Function

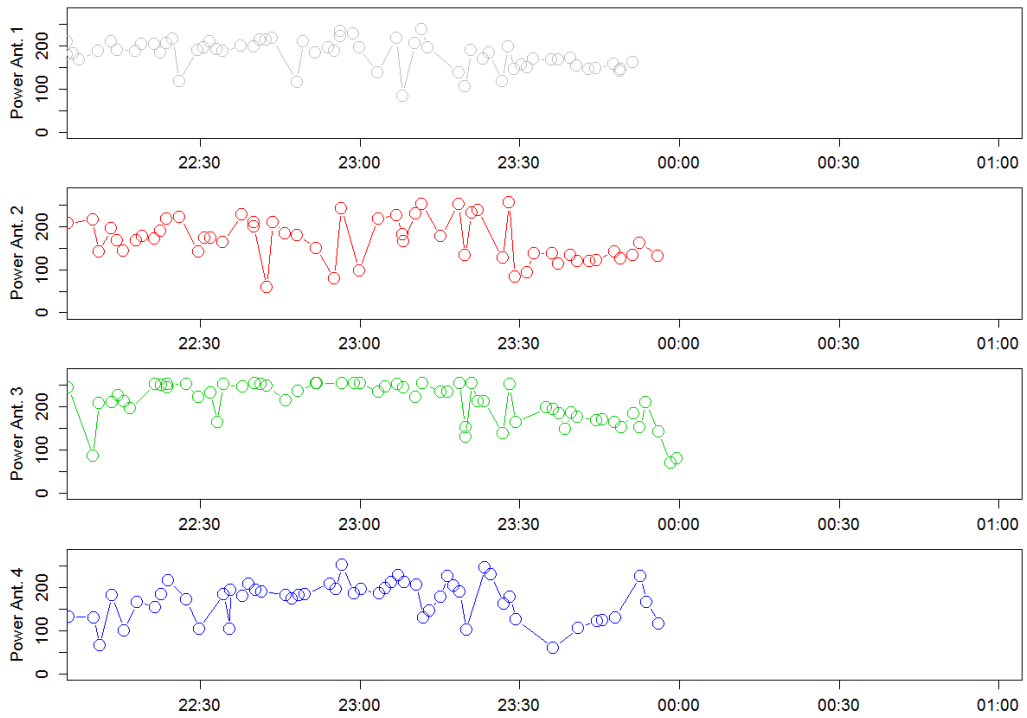
Now we can apply the zoom function to tag 155. Note by calling zoom(tag_155), we are applying the zoom function to the detection data of tag ID 155. Before applying the zoom function, you need to point R to the screen (antenna) that you are interested in zooming in on. Here, we zoom in on antenna 3, and we can only meaningfully interact with a single screen/antenna at a time. Note that you can keep zooming in further, but each time you wish to zoom, you need to specify both lines of the following code, to specify the screen you want to zoom in on and call the zoom() function. Otherwise the plots will go blank. In order to zoom, you move the cursor over the panel you've specified and specify the boundaries of the new (zoomed) plot by clicking on the plot four times, in four different areas that encapsulate the data you want to view. Essentially, you will make a box around the area you want to zoom in on.

```
screen(3, FALSE)
zoom(tag_155)
```



Zoom in a second time (requires defining the boundaries again):

```
screen(3, FALSE)
zoom(tag_155)
```



Measure values from a plot

Once we've sufficiently zoomed in to the plot, we can use the following function to measure departure timing. This function can be modified to measure other parameters of interest.

```
get_data <- function(x) {

 Tag <- readline(prompt="(1) Please enter the tag ID: " )

 which_screen <- readline(prompt="(2) If you are using multiple screens,
which screen do you wish to measure the data from? " )

 screen(as.numeric(as.character(which_screen)), FALSE) #Note we need to
specify erase = FALSE.  Default is true.  If the FALSE statement is not
included, the identify function will still work functionally, but will give
erroneous values.

 readline(prompt="(3) Please select the value of interest (press enter first)
" )

 row_numbers <- identify(x$Date_Time, x$Power, n=1, plot=FALSE)

 values <- data.frame(matrix(1:3,1)) #Note: create a dataframe to store data;
3 columns, 1 row.

 names(values) <- c("Tag_id", "Power", "Date_Time") #Note: supply the names
for the columns created above.

 values$Tag_id <- Tag

 values$Power <- x$Power[row_numbers]

 values$Date_Time <- x$Date_Time[row_numbers]

 return(values)

}
```

We can then apply the measuring function to the data. You will be prompted to enter 1) the tag ID, 2) which screen (antenna) you want to extract the time from, and 3) press enter, then use your cursor to select the point on the plot that you want to extract the time stamp for. Looking at the last plot, I chose to put my cursor in panel 3 on the last detection (green). Picking the last observation will give you the time of last detection. If you want departure time, you could instead pick the point on the plot where the bird appears to become active and depart, which may be before the final detection (this may be more obvious for some individuals than others).

```
departure_time <- NULL
departure_time <- rbind(departure_time, get_data(tag_155))
departure_time <- rbind(departure_time, get_data(tag_155)) # can do it again
to get info from a separate screen
> departure_time # print the selected time
```

For this example, departure_time gave me the following output, which could then be written to file, if desired:

```
Tag_id          Power           Date_Time
155             80              2010-10-03 23:59:30
```

```
write.csv(departure_time, file = "DepartureTime_Tag155.csv", row.names =
FALSE) # write to file, if you want


close.screen(all=T) # close all screens - will have to start plots over
again.
```