Analysis of octopus behavior from motion-activated camera trap data

Jefferson Humbert and Kirt L Onthank

11/4/2021

Contents

1 Loading required libraries for the analysis

Included in these packages is a custom package that Kirt Onthank authored for easily plotting circular time histograms titled "CircularTimeHistogram". This can be installed from github. The code for installing that package is in this chunk.

```
library(remotes)
#install_github("KirtOnthank/CircularTimeHistogram")
library(CircularTimeHistogram)
library(googlesheets4)
library(googledrive)
library(lubridate)
library(maptools)
library(circular)
library(plotrix)
library(perm)
library(vcd)
library(corrplot)
library(png)
```

2 Reading in data and cleaning it up

```
events=read.csv("event_log.csv",header=T,colClasses=c("integer",rep("character",12)))
colnames(events)=c("Deployment #","Date","Time","OctoID","Bottle ID","Event_type",
                   "Species","Behavior","Notes","1st Analyst ID","2nd Analyst ID",
                   "3rd Analyst ID","4th Analyst ID")
```

Cleaning out unfilled rows.

```
events=events[complete.cases(events$Date),]
events=events[complete.cases(events$Behavior),]
```

Reformatting time to POSIX class to make it easier to calculate times. There seems to be two different formats for dates used in the dataset, so I run this command twice.

```
events$realtime_original=
  strptime(paste(events$Date,events$Time),format="%m/%d/%Y %H:%M:%S")
events$realtime_original[is.na(events$realtime_original)]=
  strptime(paste(events$Date[is.na(events$realtime_original)],
                 events$Time[is.na(events$realtime_original)]),format="%Y/%m/%d %H:%M:%S")
```

2.1 Fixing times

Some of the cameras had the incorrect time onboard, so timestamps on the pictures are incorrect. We fix this by using the onset daylight and dark, along with the calculated sunrise and sunset at these locations to correct the times of these deployments.

```
offsets=read.csv("time_offsets.csv",header=T,
                 colClasses=c("integer","Date",rep("POSIXct",10),"integer"))
colnames(offsets)=c("Deployment","Date","H2O Sunrise","H2O Sunset","RealTime Sunrise",
                    "RealTime Sunset","light duration underwater","light duration above water",
                    "difference","half difference","real H2O sunrise","offset","offset_min")
```

```
offsets=offsets[complete.cases(offsets),]

offset.tab=aggregate(offset_min~Deployment,data=offsets,FUN="mean")

events$realtime=events$realtime_original

for (i in 1:nrow(offset.tab)){
  events$realtime[events$`Deployment #`==offset.tab$Deploymen[i]]=
    events$realtime_original[events$`Deployment #`==offset.tab$Deploymen[i]]+
    (offset.tab$offset_min[i]*60)
}
```

3 Den Use Dynamics

3.1 Visitation and excursion durations

First we extract data for how long octopuses visited the bottles, and how long they left when they left the bottles and camera field of view ### Making the visitation dataframe First we make datasets that can store

all of that data. This include dataframes for two subsequent arrivals (badarrival) or departures (baddepar) of the same octopus.

```
visit=data.frame(deployment=as.numeric(NA),
           octoID=as.character(NA),
           time=as.POSIXct(NA),
           duration=as.numeric(NA))

excursion=data.frame(deployment=as.numeric(NA),
           octoID=as.character(NA),
           time=as.POSIXct(NA),
           duration=as.numeric(NA))

badarrival=data.frame(deployment=as.numeric(NA),
           octoID=as.character(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))

baddepar=data.frame(deployment=as.numeric(NA),
           octoID=as.character(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))
```

3.1.1 Extracting visitation durations

Next, we fill those datasets by finding arrivals followed by departures for visitations, and take the duration, or departures followed by arrivals for excursions and take those durations.

```
deployments=unique(events$`Deployment #`)


for (i in 1:length(deployments)){
  arrivals=events$realtime[events$Species=="octopus_rubescens"&
                     events$Event_type=="resident non-interaction"&
                     events$Behavior=="frame_arrival"&
                     events$`Deployment #`==deployments[i]]

  arrivals.ID=events$OctoID[events$Species=="octopus_rubescens"&
                     events$Event_type=="resident non-interaction"&
                     events$Behavior=="frame_arrival"&
                     events$`Deployment #`==deployments[i]]

  departures=events$realtime[events$Species=="octopus_rubescens"&
                     events$Event_type=="resident non-interaction"&
                     events$Behavior=="frame_departure"&
                     events$`Deployment #`==deployments[i]]

  departures.ID=events$OctoID[events$Species=="octopus_rubescens"&
                     events$Event_type=="resident non-interaction"&
                     events$Behavior=="frame_departure"&
                     events$`Deployment #`==deployments[i]]
```

```
depar.tags=c(rep("arrive",length(arrivals)),rep("depart",length(departures)))

deploy.sub=data.frame(time=c(arrivals,departures),octoID=c(arrivals.ID,departures.ID),
                      type=depar.tags)
deploy.sub=deploy.sub[order(deploy.sub$time),]

octos.present=unique(deploy.sub$octoID)

for (j in 1:length(octos.present)){
  if(sum(deploy.sub$octoID==octos.present[j],na.rm=T)>1){
    deploy.single.octo=
      deploy.sub[deploy.sub$octoID==octos.present[j],] #selecting only one octopus at a time

    #Finding which arrivals are immediately followed by a departure
    visit.depar=which(deploy.single.octo$type[1:(nrow(deploy.single.octo)-1)]=="arrive"&
                        deploy.single.octo$type[2:nrow(deploy.single.octo)]=="depart")
    visit.dura=difftime(deploy.single.octo$time[visit.depar+1],
                        deploy.single.octo$time[visit.depar],units = "min")
    visit.stub=data.frame(deployment=rep(deployments[i],length(visit.dura)),
               octoID=rep(octos.present[j],length(visit.dura)),
               time=deploy.single.octo$time[visit.depar],
               duration=as.numeric(visit.dura))
    visit=rbind(visit,visit.stub)

    # Finding which departures are immediately followed by an arrival
    excur.depar=which(deploy.single.octo$type[1:(nrow(deploy.single.octo)-1)]=="depart"&
                        deploy.single.octo$type[2:nrow(deploy.single.octo)]=="arrive")
    excur.dura=difftime(deploy.single.octo$time[excur.depar+1],
                        deploy.single.octo$time[excur.depar],units = "min")
    excur.stub=data.frame(deployment=rep(deployments[i],length(excur.dura)),
               octoID=rep(octos.present[j],length(excur.dura)),
               time=deploy.single.octo$time[excur.depar],
               duration=as.numeric(excur.dura))
    excursion=rbind(excursion,excur.stub)

    # Finding arrivals followed immediately by another arrival
    #(This should not be possible, so this is error locating)
    badarrival.depar=which(deploy.single.octo$type[1:(nrow(deploy.single.octo)-1)]=="arrive"&
                             deploy.single.octo$type[2:nrow(deploy.single.octo)]=="arrive")
    badarrival.dura=difftime(deploy.single.octo$time[badarrival.depar+1],
                             deploy.single.octo$time[badarrival.depar],units = "min")
    badarrival.stub=data.frame(deployment=rep(deployments[i],length(badarrival.dura)),
               octoID=rep(octos.present[j],length(badarrival.dura)),
               time1=deploy.single.octo$time[badarrival.depar],
               time2=deploy.single.octo$time[badarrival.depar+1],
               duration=as.numeric(badarrival.dura))
    badarrival=rbind(badarrival,badarrival.stub)

    baddepar.depar=which(deploy.single.octo$type[1:(nrow(deploy.single.octo)-1)]=="depart"&
                           deploy.single.octo$type[2:nrow(deploy.single.octo)]=="depart")
    baddepar.dura=difftime(deploy.single.octo$time[baddepar.depar+1],
                           deploy.single.octo$time[baddepar.depar],units = "min")
    baddepar.stub=data.frame(deployment=rep(deployments[i],length(baddepar.dura)),
```

```
                  octoID=rep(octos.present[j],length(baddepar.dura)),
                  time1=deploy.single.octo$time[baddepar.depar],
                  time2=deploy.single.octo$time[baddepar.depar+1],
                  duration=as.numeric(baddepar.dura))
      baddepar=rbind(baddepar,baddepar.stub)

    }
  }
}

visit=visit[complete.cases(visit),]
excursion=excursion[complete.cases(excursion),]
```

3.1.2 Adding Day/Night to the visitation data based on sunrise and sunset times.

We also want to look at the differences is how octopuses are behaving between night and day. Therefore, we added if each visitation or excursion occurred during the night or the day. We defined visitations or excursions as happening during the day if they began anytime between sunrise and sunset and determined for the specific day it occurred and at the specific location of Driftwood Park using the sunriset() function in the maptools package. Night, therefore, was defined as any occurring between sunset and sunrise.

```
driftwood=matrix(c(-122.6396394,48.1639127), nrow=1)
bay=SpatialPoints(driftwood, proj4string=CRS("+proj=longlat +datum=WGS84"))

visit$daynight="night"
excursion$daynight="night"
visit$day_expected=0
excursion$day_expected=0

for (i in 1:nrow(visit)){
  sunrise=sunriset(bay, as.POSIXct(visit$time[i]), direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(visit$time[i]), direction="sunset", POSIXct.out=TRUE)$time
  if (visit$time[i]>sunrise&visit$time[i]<sunset) {
    visit$daynight[i]="day"
  }
  visit$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}

for (i in 1:nrow(excursion)){
  sunrise=sunriset(bay, as.POSIXct(excursion$time[i]), direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(excursion$time[i]), direction="sunset", POSIXct.out=TRUE)$time
  if (excursion$time[i]>sunrise&excursion$time[i]<sunset) {
    excursion$daynight[i]="day"
  }
  excursion$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}
```

What was the mean visitation duration?

```
mean(visit$duration)
```

```
## [1] 264.5949
```

What was the mean excursion duration?

```
mean(excursion$duration)
```

```
## [1] 3.022222
```

3.1.3 Adding day night and probabilities to each entry in the event log

Next, we add day or night categories for each event based on when it occurred. Finally, for statistical analysis we need to find the expected probability of events occurring during the day or night. Because the periods of night and day are not equal length, we would not expect an equal distribution of randomly occuring events between night and day. Therefore, for each event recorded in the event log we are calculating the likelihood of that event happening during the day be calculating the proportion of the daylight hours (from sunrise to sunset) of a whole 24 hour period. Again, this is calculated for the specific location of Driftwood Park and for the specific day on which the event occurred.

```
events$daynight="night"
events$day_expected=0

for (i in 1:nrow(events)){
  sunrise=sunriset(bay, as.POSIXct(events$realtime[i]), direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(events$realtime[i]), direction="sunset", POSIXct.out=TRUE)$time
  if (events$realtime[i]>sunrise&events$realtime[i]<sunset) {
    events$daynight[i]="day"
  }
  events$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}
```

3.2 Day versus Night Analysis

3.2.1 octopus departures: day vs night

We first looked at octopus departures from the frame. We decided that frame departures was a good proxy for activity, as this represents either a resident leaving their "home" bottle den, or a non-resident coming and going.

```
depart.daynight=as.numeric(table(events$daynight[events$Behavior=="frame_departure"&
                                                 events$Species=="octopus_rubescens"]))

depart.probs=c(
mean(events$day_expected[events$Behavior=="frame_departure"&
                           events$Species=="octopus_rubescens"]),
1-mean(events$day_expected[events$Behavior=="frame_departure"&
                             events$Species=="octopus_rubescens"])
)

depart.chi=chisq.test(depart.daynight,p=depart.probs)
depart.chi
```

```
##
##  Chi-squared test for given probabilities
```

```
##
## data:  depart.daynight
## X-squared = 6.0401, df = 1, p-value = 0.01398
```

3.2.2 octopus den fortification: day vs. night

Next, we looked at fortification behavior in the day versus in the night. Fortification is when an octopus pulls material over the entrance of their end, effectively closing themselves inside.

```
fort.daynight=as.numeric(table(events$daynight[events$Behavior=="fortify"&
                                                 events$Species=="octopus_rubescens"]))

fort.probs=c(
mean(events$day_expected[events$Behavior=="fortify"&
                           events$Species=="octopus_rubescens"]),
1-mean(events$day_expected[events$Behavior=="fortify"&
                             events$Species=="octopus_rubescens"])
)

fort.chi=chisq.test(fort.daynight,p=fort.probs)
fort.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  fort.daynight
## X-squared = 147.95, df = 1, p-value < 2.2e-16
```

3.2.3 octopus periscope: day vs night

Periscoping is when an octopus reaches it's eyes outside of the den, but leaves it's arms and body inside.

```
periscope.daynight=as.numeric(table(events$daynight[events$Behavior=="periscope"&
                                                      events$Species=="octopus_rubescens"]))

peri.probs=c(
mean(events$day_expected[events$Behavior=="periscope"&
                           events$Species=="octopus_rubescens"]),
1-mean(events$day_expected[events$Behavior=="periscope"&
                             events$Species=="octopus_rubescens"])
)

peri.chi=chisq.test(periscope.daynight,p=peri.probs)
peri.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  periscope.daynight
## X-squared = 9.2429, df = 1, p-value = 0.002364
```

3.2.4 octopus bottle enter and exit: day vs night

Another good proxy we thought for general octopus activity would be entering or exiting the bottle den. So, we are also looking at that between day and night to see if there is a difference in frequency.

```
enterexit.daynight=as.numeric(table(events$daynight[events$Behavior=="enter_bottle"&
                                                    events$Species=="octopus_rubescens"|
                                                    events$Behavior=="exit_bottle"&
                                                    events$Species=="octopus_rubescens"]))

enterexit.probs=c(
mean(events$day_expected[events$Behavior=="enter_bottle"&
                           events$Species=="octopus_rubescens"|
                           events$Behavior=="exit_bottle"&
                           events$Species=="octopus_rubescens"]),
1-mean(events$day_expected[events$Behavior=="enter_bottle"&
                             events$Species=="octopus_rubescens"|
                             events$Behavior=="exit_bottle"&
                             events$Species=="octopus_rubescens"])
)

enterexit.chi=chisq.test(enterexit.daynight,p=enterexit.probs)
enterexit.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  enterexit.daynight
## X-squared = 4.6117, df = 1, p-value = 0.03176
```

3.2.5 Time histogram figure

```
moon=readPNG("moon.png")
sun=readPNG("sun.png")
svg(filename="Figure2.svg")
par(fig=c(0,0.5,0.5,1))
DayHist(events$realtime[events$Species=="octopus_rubescens"&
                          events$Behavior=="fortify"],
        date="07/10/2021",hist.zoom=4,
        night.col="lightskyblue3")
rasterImage(moon,-0.15,0.6,0.15,0.9)
rasterImage(sun,-0.15,-0.6,0.15,-0.9)
mtext("Fortification",side=3,line=1.5,cex=1.5)
mtext("A",side=3,line=1.5,cex=2,adj=0)

if(fort.chi$p.value>0.0001){
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
                                 events$Behavior=="fortify",]),
              " p=",format(signif(fort.chi$p.value,1),scientific = F),sep=""),
      side=3,line=0.6,cex=1)
} else{
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
```

```
                                    events$Behavior=="fortify",]),
              " p<0.0001",sep=""),
      side=3,line=0.6,cex=1)
}

par(fig=c(0.5,1,0.5,1),new=T)
DayHist(events$realtime[events$Species=="octopus_rubescens"&
                          events$Behavior=="frame_departure"],
        date="07/10/2021",hist.zoom=8,
        night.col="lightskyblue3")
rasterImage(moon,-0.15,0.6,0.15,0.9)
rasterImage(sun,-0.15,-0.6,0.15,-0.9)
mtext("B",side=3,line=1.5,cex=2,adj=0)
mtext("Departures",side=3,line=1.5,cex=1.5)

if(depart.chi$p.value>0.0001){
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
                                    events$Behavior=="frame_departure",]),
              " p=",format(signif(depart.chi$p.value,1),scientific = F),sep=""),
      side=3,line=0.6,cex=1)
} else{
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
                                    events$Behavior=="frame_departure",]),
              " p<0.0001",sep=""),
      side=3,line=0.6,cex=1)
}

par(fig=c(0,0.5,0,0.5),new=T)
DayHist(events$realtime[events$Species=="octopus_rubescens"&
                          events$Behavior=="periscope"],
        date="07/10/2021",hist.zoom=4.5,
        night.col="lightskyblue3")
rasterImage(moon,-0.15,0.6,0.15,0.9)
rasterImage(sun,-0.15,-0.6,0.15,-0.9)
mtext("C",side=3,line=1.5,cex=2,adj=0)
mtext("Periscoping",side=3,line=1.5,cex=1.5)

if(peri.chi$p.value>0.0001){
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
                                    events$Behavior=="periscope",]),
              " p=",format(signif(peri.chi$p.value,1),scientific = F),sep=""),
      side=3,line=0.6,cex=1)
} else{
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
                                    events$Behavior=="periscope",]),
              " p<0.0001",sep=""),
      side=3,line=0.6,cex=1)
}


par(fig=c(0.5,1,0,0.5),new=T)
DayHist(events$realtime[events$Species=="octopus_rubescens"&
                          events$Behavior=="enter_bottle"|
```

```
                          events$Species=="octopus_rubescens"&
                          events$Behavior=="exit_bottle"],
        date="07/10/2021",hist.zoom=4.5,
        night.col="lightskyblue3")
rasterImage(moon,-0.15,0.6,0.15,0.9)
rasterImage(sun,-0.15,-0.6,0.15,-0.9)
mtext("D",side=3,line=1.5,cex=2,adj=0)
mtext("Enter/exit bottle",side=3,line=1.5,cex=1.5)
if(enterexit.chi$p.value>0.0001){
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
                                 events$Behavior=="enter_bottle"|
                                 events$Species=="octopus_rubescens"&
                                 events$Behavior=="exit_bottle",]),
              " p=",format(signif(enterexit.chi$p.value,1),scientific = F),sep=""),
      side=3,line=0.6,cex=1)
} else{
  mtext(paste("n=",nrow(events[events$Species=="octopus_rubescens"&
                                 events$Behavior=="enter_bottle"|
                                 events$Species=="octopus_rubescens"&
                                 events$Behavior=="exit_bottle",]),
              " p<0.0001",sep=""),
      side=3,line=0.6,cex=1)
}

dev.off()
```

```
## pdf
##   2
```

This is bash code to convert the svg to png that can be displayed in the RMarkdown pdf.

```
cairosvg Figure2.svg -o Figure2.png -d 100
```

And this is bash code to convert the svg to eps for publication.

```
inkscape Figure2.svg -o Figure2.eps --export-ignore-filters --export-ps-level=3
```

4 Conspecific interactions

Next up, we look at interactions between octopuses. Specifically we want to look at frequency and duration of non-resident octopus visits. Non-resident octopuses are defined as those that do not enter a bottle in the camera field of view at any time during the deployment.

4.1 Resident/Non-resident interaction

4.1.1 Finding times when resident octopuses are present and absent

We would like to examine if the presence or absence of a resident octopus changes the behavior of non-resident octopuses. In order to do this, we need to look at when resident octopuses are present and absent.

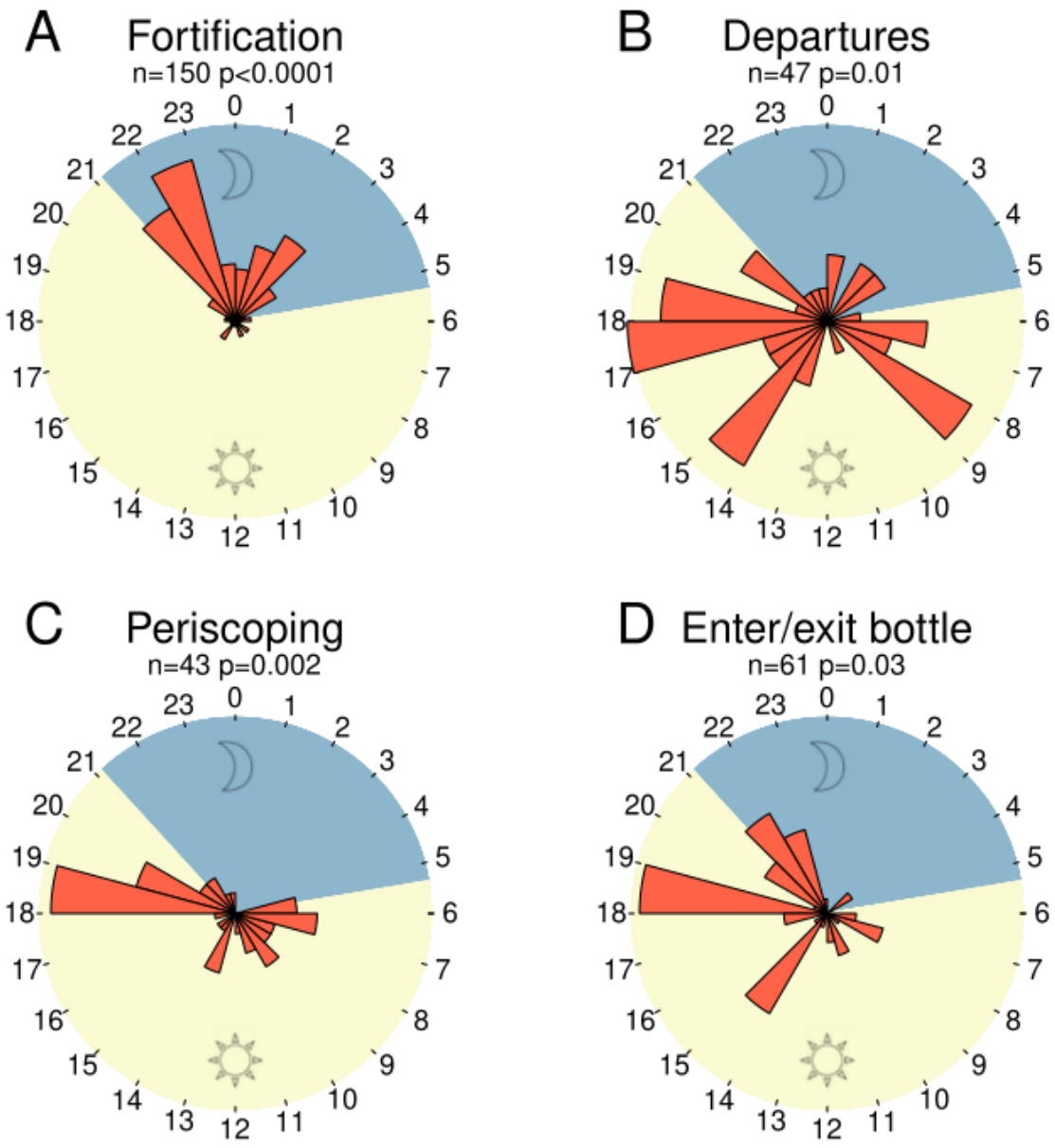

Figure 1: Radial histograms representing the absolute frequencies of hourly daily activities for O. rubescens recorded by marine camera traps throughout the day.

```
residents.present=events[events$Behavior=="deploy_start"|
         events$Species=="octopus_rubescens"&
         events$Behavior=="frame_arrival"&
         events$Event_type=="resident non-interaction"|
         events$Species=="octopus_rubescens"&
         events$Behavior=="frame_departure"&
         events$Event_type=="resident non-interaction"|
         events$Species=="octopus_rubescens"&
         events$Behavior=="deploy_start"&
         events$Event_type=="resident non-interaction"|
         events$Behavior=="deploy_end",
         ]
events$Behavior[events$Behavior=="second_departure"]="frame_departure"
```

Next, I make sure that the entries in this dataframe are in chronological order within each deployment.

```
for (i in 1:length(deployments)){
  temp=residents.present[residents.present$`Deployment #`==deployments[i],]
  temp=temp[order(temp$realtime),]
  residents.present[residents.present$`Deployment #`==deployments[i],]=temp
}
```

Now, I add a column to the dataframe called "number" that will be a tally of how many octopuses are in the frame. I will start each deployment with the number at 0, add an octopus for each frame arrival and substract one octopus for each frame departure.

```
residents.present$number=0

for (i in 1:length(deployments)){
  behaviors=residents.present$Behavior[residents.present$`Deployment #`==deployments[i]]
  present=rep(0,length(behaviors))
  for (j in 2:length(behaviors)){
    if (behaviors[j]=="frame_arrival"){
      present[j]=present[j-1]+1
    }
    if (behaviors[j]=="frame_departure"){
      present[j]=present[j-1]-1
    }
    if (behaviors[j]=="deploy_end"){
      present[j]=present[j-1]
    }
  }
  residents.present$number[residents.present$`Deployment #`==deployments[i]]=present
}
```

The dataframe I made has the start time for each time period, but not the end. I will add the end by taking the starting time of the next time period.

```
residents.present$endtime=residents.present$realtime[1]


for (i in 1:length(deployments)){
```

```
  #All of the times excepts the last (which is the end of deployment)
  residents.present$endtime[head(which(residents.present$`Deployment #`==deployments[i]),-1)]=
    #is replaced by all of the times except the first
    residents.present$realtime[tail(which(residents.present$`Deployment #`==deployments[i]),-1)]
}

residents.present=residents.present[!residents.present$Behavior=="deploy_end",]
```

Finding durations of each time period

```
residents.present$duration=
  as.numeric(difftime(residents.present$endtime,residents.present$realtime,units = "min"))
```

finding total time with and without octopuses

```
min.res.absent=sum(residents.present$duration[residents.present$number==0])
min.res.present=sum(residents.present$duration[residents.present$number>0])
min.res.absent
```

```
## [1] 23534.63
```

```
min.res.present
```

```
## [1] 19353.98
```

4.1.2 non-resident octopus visit duration and frequency when resident octopuses are absent versus present.

First, I make dataframes to hold my visit duration information and catalog potential bad arrivals and departure entries.

```
octo.visit=data.frame(deployment=as.numeric(NA),
           time=as.POSIXct(NA),
           duration=as.numeric(NA))

octo.badarrival=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))

octo.baddepar=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))
```

Next, I calculate the duration of each octopus visit.

```
for (i in 1:length(deployments)){
  arrivals=events$realtime[events$Species=="octopus_rubescens"&
                     events$Behavior=="frame_arrival"&
```

```
                    events$Event_type=="non-resident non-interaction"&
                    events$`Deployment #`==deployments[i]]

  departures=events$realtime[events$Species=="octopus_rubescens"&
                    events$Behavior=="frame_departure"&
                    events$Event_type=="non-resident non-interaction"&
                    events$`Deployment #`==deployments[i]]

  if (length(arrivals)>0){
  depar.tags=c(rep("arrive",length(arrivals)),rep("depart",length(departures)))

  deploy.sub=data.frame(time=c(arrivals,departures),type=depar.tags)
  deploy.sub=deploy.sub[order(deploy.sub$time),]

      #Finding which arrivals are immediately followed by a departure
      visit.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                          deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      # Now calculating the
      visit.dura=difftime(deploy.sub$time[visit.depar+1],
                          deploy.sub$time[visit.depar],units = "min")
      visit.stub=data.frame(deployment=rep(deployments[i],length(visit.dura)),
                 time=deploy.sub$time[visit.depar],
                 duration=as.numeric(visit.dura))
      octo.visit=rbind(octo.visit,visit.stub)

      # Finding arrivals followed immediately by another arrival
      #(This should not be possible, so this is error locating)
      badarrival.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                               deploy.sub$type[2:nrow(deploy.sub)]=="arrive")
      badarrival.dura=difftime(deploy.sub$time[badarrival.depar+1],
                               deploy.sub$time[badarrival.depar],units = "min")
      badarrival.stub=data.frame(deployment=rep(deployments[i],length(badarrival.dura)),
                 time1=deploy.sub$time[badarrival.depar],
                 time2=deploy.sub$time[badarrival.depar+1],
                 duration=as.numeric(badarrival.dura))
      octo.badarrival=rbind(octo.badarrival,badarrival.stub)

      baddepar.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="depart"&
                             deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      baddepar.dura=difftime(deploy.sub$time[baddepar.depar+1],
                             deploy.sub$time[baddepar.depar],units = "min")
      baddepar.stub=data.frame(deployment=rep(deployments[i],length(baddepar.dura)),
                 time1=deploy.sub$time[baddepar.depar],
                 time2=deploy.sub$time[baddepar.depar+1],
                 duration=as.numeric(baddepar.dura))
      octo.baddepar=rbind(octo.baddepar,baddepar.stub)


    }
}

octo.visit=octo.visit[complete.cases(octo.visit),]
```

Adding if resident octopuses are present or absent during each visit to my dataframe

```
octo.visit$resident="unknown"


for (i in 1:nrow(residents.present)){
  if (residents.present$number[i]>0){
    octo.visit$resident[octo.visit$deployment==residents.present$`Deployment #`[i]&
                octo.visit$time>residents.present$realtime[i]&
                octo.visit$time<residents.present$endtime[i]]="present"
  }
  if (residents.present$number[i]==0){
    octo.visit$resident[octo.visit$deployment==residents.present$`Deployment #`[i]&
                octo.visit$time>residents.present$realtime[i]&
                octo.visit$time<residents.present$endtime[i]]="absent"
  }
}

octo.visit=octo.visit[!octo.visit$resident=="unknown",]
```

Adding if each octo visit was during the day or the night

```
octo.visit$daynight="night"
octo.visit$day_expected=0

for (i in 1:nrow(octo.visit)){
  sunrise=sunriset(bay, as.POSIXct(octo.visit$time[i]),
                   direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(octo.visit$time[i]),
                  direction="sunset", POSIXct.out=TRUE)$time
  if (octo.visit$time[i]>sunrise&octo.visit$time[i]<sunset) {
    octo.visit$daynight[i]="day"
  }
  octo.visit$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}
```

4.1.3 Non-resident octopus frequency chi-square analysis

Getting distribution of observations for chi-squared test

```
octo.tab=
  as.vector(table(octo.visit$resident))
octo.tab
```

```
## [1] 28 19
```

Setting up probabilities for chi-squared test based on the proportion of time that resident octopuses were present or absent.

```
octo.probs=
    c(min.res.absent/(min.res.absent+min.res.present),
      min.res.present/(min.res.absent+min.res.present))
octo.probs
```

```
## [1] 0.5487385 0.4512615
```

Running chi-squared test

```
octo.chi=chisq.test(octo.tab,p=octo.probs)
octo.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  octo.tab
## X-squared = 0.41939, df = 1, p-value = 0.5172
```

Extracting residuals for plotting later.

```
octo.res=matrix(octo.chi$residuals,ncol=2)
colnames(octo.res)=c("resident absent","resident present")
```

4.1.4 Non-resident octopus visit duration T-test

Testing my assumptions for T-Test

```
shapiro.test(octo.visit$duration[octo.visit$resident=="present"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  octo.visit$duration[octo.visit$resident == "present"]
## W = 0.83763, p-value = 0.004248
```

```
shapiro.test(octo.visit$duration[octo.visit$resident=="absent"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  octo.visit$duration[octo.visit$resident == "absent"]
## W = 0.91957, p-value = 0.03386
```

```
bartlett.test(octo.visit$duration,octo.visit$resident)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  octo.visit$duration and octo.visit$resident
## Bartlett's K-squared = 19.155, df = 1, p-value = 1.205e-05
```

Assumptions fail, so I am using a permutation two-sample test.

```
octo.permTS=permTS(octo.visit$duration[octo.visit$resident=="absent"],
                   octo.visit$duration[octo.visit$resident=="present"],
       alternative="two.sided",method="exact.mc",control=permControl(nmc=10000))
octo.permTS
```

```
##
##  Exact Permutation Test Estimated by Monte Carlo
##
## data:  GROUP 1 and GROUP 2
## p-value = 0.0122
## alternative hypothesis: true mean GROUP 1 - mean GROUP 2 is not equal to 0
## sample estimates:
## mean GROUP 1 - mean GROUP 2
##                   -1.168202
##
## p-value estimated from 10000 Monte Carlo replications
## 99 percent confidence interval on p-value:
##  0.008392331 0.016578858
```

So, it appears that non-resident octopus visit duration is significantly different if a resident octopus is present.

Seeing what the mean difference is between non-resident visit duration when octopuses are absent versus octopuses are present.

```
aggregate(duration~resident,data=octo.visit,FUN="mean")
```

```
##   resident duration
## 1   absent 1.095833
## 2  present 2.264035
```

4.1.5 Frequency of Bottle reaching

Adding a field to the octo.visit object called "reach", and then filling it with simply if there is a reach event into

```
octo.visit$reach="unknown"

for (i in 1:nrow(octo.visit)){
  start=octo.visit$time[i]
  end=start+(octo.visit$duration[i]*60)
  happenings=events$Behavior[events$realtime>=start&
                               events$realtime<=end&
                               events$`Deployment #`==octo.visit$deployment[i]]
  if(sum(grepl("reach_inside_inhabited_bottle|reach_inside_uninhabited_bottle",
               happenings))>0){
    octo.visit$reach[i]="reach"
  } else {
    octo.visit$reach[i]="no_reach"
  }
}
```

Frequency of reaching and no reaching inside a bottle when a resident is present and absent.

```
table(octo.visit$resident,octo.visit$reach)
```

```
##
##           no_reach reach
##   absent        16    12
##   present        6    13
```

Getting distribution of reach and no reach when resident octopus is present for chi-squared test

```
reach.tab=
  table(octo.visit$reach[octo.visit$resident=="present"])
reach.tab
```

```
##
## no_reach    reach
##        6       13
```

Setting up probabilities for chi-squared test. I am doing this such that the expected is the ratio when resident is absent.

```
absent.tab=table(octo.visit$reach[octo.visit$resident=="absent"])

reach.probs=c(absent.tab[1]/sum(absent.tab),absent.tab[2]/sum(absent.tab))
reach.probs
```

```
##  no_reach     reach
## 0.5714286 0.4285714
```

Running chi-squared test

```
reach.chi=chisq.test(reach.tab,p=reach.probs)
reach.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  reach.tab
## X-squared = 5.0702, df = 1, p-value = 0.02434
```

```
reach.res=matrix(reach.chi$residuals,ncol=2)
colnames(reach.res)=c("resident absent","resident present")
reach.res
```

```
##      resident absent resident present
## [1,]       -1.474087         1.702129
```

5 Interspecific Interaction Analysis

Turning all rockfish species into just "rockfish". I am doing this down here so I don't break your code above that relies on your sorting with different rockfish. If you change that, we can put this earlier in the code.

```
events$Species[grep("rockfish",events$Species)]="rockfish"
```

Now I am taking a quick look at how many time each species appears in the data.

```
table(events$Species[events$Behavior=="frame_arrival"])
```

```
##
##   buffalo sculpin       cancer crab           chimera    dungeness crab
##                19                10                 4                22
##          flatfish       helmet_crab    kelp greenling           lingcod
##                21                 3               302               215
## octopus_rubescens painted greenling        pile_perch     red rock crab
##                78                60                30               390
##          rockfish   sailfin sculpin      sea_cucumber          sea_star
##               640                11                 3                 3
##     spiny dogfish           unknown            urchin          wolf eel
##                 7                 2                 1                 7
```

It looks like we have enough data to do this analysis on rockfish, red rock crabs, kelp greenling, and lingcod. I think everything else doesn't have enough sightings to be worthwile.

5.1 Finding times when octopuses are present or absent

We are interested in how the presence of an octopus near the den affects the behavior of other commonly seen animals. In particularly we are curious is visitation frequency or duration is different when an octopus is present versus absent. In order to investigate this, we first need to build a dataset of when octopuses are present and absent.

```
events$Behavior[events$Behavior=="second_departure"]="frame_departure"
events$Behavior[events$Behavior=="second_arrival"]="frame_arrival"

octos.present=events[events$Behavior=="deploy_start"|
         events$Species=="octopus_rubescens"&
         events$Behavior=="frame_arrival"|
         events$Species=="octopus_rubescens"&
         events$Behavior=="frame_departure"|
         events$Species=="octopus_rubescens"&
         events$Behavior=="deploy_start"|
         events$Behavior=="deploy_end",
         ]
```

Next, I make sure that the entries in this dataframe are in chonological order within each deployment.

```
for (i in 1:length(deployments)){
  temp=octos.present[octos.present$`Deployment #`==deployments[i],]
  temp=temp[order(temp$realtime),]
  octos.present[octos.present$`Deployment #`==deployments[i],]=temp
}
```

Now, I add a column to the dataframe called "number" that will be a tally of how many octopuses are in the frame. I will start each deployment with the number at 0, add an octopus for each frame arrival and substract one octopus for each frame departure.

```
octos.present$number=0

for (i in 1:length(deployments)){
  behaviors=octos.present$Behavior[octos.present$`Deployment #`==deployments[i]]
  present=rep(0,length(behaviors))
  for (j in 2:length(behaviors)){
    if (behaviors[j]=="frame_arrival"){
      present[j]=present[j-1]+1
    }
    if (behaviors[j]=="frame_departure"){
      present[j]=present[j-1]-1
    }
    if (behaviors[j]=="deploy_end"){
      present[j]=present[j-1]
    }
  }
  octos.present$number[octos.present$`Deployment #`==deployments[i]]=present
}
```

The dataframe I made has the start time for each time period, but not the end. I will add the end by taking the starting time of the next time period.

```
octos.present$endtime=octos.present$realtime[1]


for (i in 1:length(deployments)){
  octos.present$endtime[head(which(octos.present$`Deployment #`==deployments[i]),-1)]=
    #All of the times excepts the last (which is the end of deployment)
    octos.present$realtime[tail(which(octos.present$`Deployment #`==deployments[i]),-1)]
    #is replaced by all of the times except the first
}

octos.present=octos.present[!octos.present$Behavior=="deploy_end",]
```

Finding durations of each time period

```
octos.present$duration=as.numeric(difftime(octos.present$endtime,
                                           octos.present$realtime,units = "min"))
```

finding total time with and without octopuses

```
min.absent=sum(octos.present$duration[octos.present$number==0])
min.present=sum(octos.present$duration[octos.present$number>0])
min.absent
```

```
## [1] 26916.37
```

```
min.present
```

```
## [1] 15972.25
```

Finding the proportion of times when octopuses are present, in which there is more than one octopus present.

```
sum(octos.present$duration[octos.present$number>=1])
```

```
## [1] 15972.25
```

```
sum(octos.present$duration[octos.present$number>1])
```

```
## [1] 6535.933
```

What is the proportion of total time when at least one octopus is present?

```
sum(octos.present$duration[octos.present$number>1])/
  sum(octos.present$duration[octos.present$number>=1])
```

```
## [1] 0.4092055
```

Finding total number of conspecific interactions

```
sum(octos.present$number>=2)
```

```
## [1] 43
```

5.2 Kelp Greenling Analysis

5.2.1 Kelp Greenling visitation frequency with octopuses absent vs. present.

In this chunk I am tallying kelp greenling arrivals when octopuses are present versus absent.

```
kelpies=numeric()

for (i in 1:nrow(octos.present)){
  kelpies[i]=sum(events$Species=="kelp greenling"&
    events$Behavior=="frame_arrival"&
    events$`Deployment #`==octos.present$`Deployment #`[i]&
    events$realtime>octos.present$realtime[i]&
    events$realtime<octos.present$endtime[i])
}

kelpies.absent=sum(kelpies[octos.present$number==0])
kelpies.present=sum(kelpies[octos.present$number>0])
kelpies.absent
```

```
## [1] 172
```

```
kelpies.present
```

```
## [1] 130
```

Next I perform a chi square test to see if the kelp greenling visitation frequency is significantly different that random.Here, I am using the probability in each category (absent vs. present) as the proportion of time octopuses were absent versus present.

```
probs=c(min.absent/(min.absent+min.present),min.present/(min.absent+min.present))

chisq.test(c(kelpies.absent,kelpies.present),p=probs)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(kelpies.absent, kelpies.present)
## X-squared = 4.3544, df = 1, p-value = 0.03691
```

5.2.2 Kelp greenling visitation duration when octopuses are absent versus present.

First, I make dataframes to hold my visit duration information and catalog potential bad arrivals and departure entries.

```
kelpie.visit=data.frame(deployment=as.numeric(NA),
           time=as.POSIXct(NA),
           duration=as.numeric(NA))

kelpie.badarrival=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))

kelpie.baddepar=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))
```

Next, I calculate the duration of each kelp greenling visit.

```
for (i in 1:length(deployments)){
  arrivals=events$realtime[events$Species=="kelp greenling"&
                     events$Behavior=="frame_arrival"&
                     events$`Deployment #`==deployments[i]]

  departures=events$realtime[events$Species=="kelp greenling"&
                     events$Behavior=="frame_departure"&
                     events$`Deployment #`==deployments[i]]

  if (length(arrivals)>0){
  depar.tags=c(rep("arrive",length(arrivals)),rep("depart",length(departures)))

  deploy.sub=data.frame(time=c(arrivals,departures),type=depar.tags)
  deploy.sub=deploy.sub[order(deploy.sub$time),]

      #Finding which arrivals are immediately followed by a departure
      visit.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
```

```
                          deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      # Now calculating the
      visit.dura=difftime(deploy.sub$time[visit.depar+1],deploy.sub$time[visit.depar],units = "min")
      visit.stub=data.frame(deployment=rep(deployments[i],length(visit.dura)),
                 time=deploy.sub$time[visit.depar],
                 duration=as.numeric(visit.dura))
      kelpie.visit=rbind(kelpie.visit,visit.stub)

      # Finding arrivals followed immediately by another arrival
      #(This should not be possible, so this is error locating)
      badarrival.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                               deploy.sub$type[2:nrow(deploy.sub)]=="arrive")
      badarrival.dura=difftime(deploy.sub$time[badarrival.depar+1],
                               deploy.sub$time[badarrival.depar],units = "min")
      badarrival.stub=data.frame(deployment=rep(deployments[i],length(badarrival.dura)),
                 time1=deploy.sub$time[badarrival.depar],
                 time2=deploy.sub$time[badarrival.depar+1],
                 duration=as.numeric(badarrival.dura))
      kelpie.badarrival=rbind(kelpie.badarrival,badarrival.stub)

      baddepar.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="depart"&
                             deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      baddepar.dura=difftime(deploy.sub$time[baddepar.depar+1],
                             deploy.sub$time[baddepar.depar],units = "min")
      baddepar.stub=data.frame(deployment=rep(deployments[i],length(baddepar.dura)),
                 time1=deploy.sub$time[baddepar.depar],
                 time2=deploy.sub$time[baddepar.depar+1],
                 duration=as.numeric(baddepar.dura))
      kelpie.baddepar=rbind(kelpie.baddepar,baddepar.stub)


    }
}


kelpie.visit=kelpie.visit[complete.cases(kelpie.visit),]
```

Adding if octopuses are present or absent during each visit to my dataframe

```
kelpie.visit$octopus="unknown"


for (i in 1:nrow(octos.present)){
  if (octos.present$number[i]>0){
    kelpie.visit$octopus[kelpie.visit$deployment==octos.present$`Deployment #`[i]&
                 kelpie.visit$time>octos.present$realtime[i]&
                 kelpie.visit$time<octos.present$endtime[i]]="present"
  }
  if (octos.present$number[i]==0){
    kelpie.visit$octopus[kelpie.visit$deployment==octos.present$`Deployment #`[i]&
                 kelpie.visit$time>octos.present$realtime[i]&
                 kelpie.visit$time<octos.present$endtime[i]]="absent"
  }
```

```
}

kelpie.visit=kelpie.visit[!kelpie.visit$octopus=="unknown",]
```

Adding if each kelpie visit was during the day or the night

```
kelpie.visit$daynight="night"
kelpie.visit$day_expected=0

for (i in 1:nrow(kelpie.visit)){
  sunrise=sunriset(bay, as.POSIXct(kelpie.visit$time[i]),
                   direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(kelpie.visit$time[i]),
                  direction="sunset", POSIXct.out=TRUE)$time
  if (kelpie.visit$time[i]>sunrise&kelpie.visit$time[i]<sunset) {
    kelpie.visit$daynight[i]="day"
  }
  kelpie.visit$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}
```

Getting distribution of observations for chi-squared test

```
kelpie.tab=
  as.vector(table(kelpie.visit$octopus))
kelpie.tab
```

```
## [1] 164 127
```

Setting up probabilities for chi-squared test

```
kelpie.probs=
  c(min.absent/(min.absent+min.present),min.present/(min.absent+min.present))
kelpie.probs
```

```
## [1] 0.6275877 0.3724123
```

Running chi-squared test

```
kelpie.chi=chisq.test(kelpie.tab,p=kelpie.probs)
kelpie.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  kelpie.tab
## X-squared = 5.102, df = 1, p-value = 0.0239
```

Putting residuals into object for later plotting

```
kelpie.res=matrix(kelpie.chi$residuals,ncol=2)
colnames(kelpie.res)=c("octopus absent","octopus present")
```

Getting Kelpie observations per hour

```
kelpie.tab[1]/(min.absent/60/24)
```

```
## [1] 8.773844
```

```
kelpie.tab[2]/(min.present/60/24)
```

```
## [1] 11.44986
```

Testing my assumptions for T-Test

```
shapiro.test(kelpie.visit$duration[kelpie.visit$octopus=="present"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  kelpie.visit$duration[kelpie.visit$octopus == "present"]
## W = 0.57982, p-value < 2.2e-16
```

```
shapiro.test(kelpie.visit$duration[kelpie.visit$octopus=="absent"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  kelpie.visit$duration[kelpie.visit$octopus == "absent"]
## W = 0.49472, p-value < 2.2e-16
```

```
bartlett.test(kelpie.visit$duration,kelpie.visit$octopus)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  kelpie.visit$duration and kelpie.visit$octopus
## Bartlett's K-squared = 133.96, df = 1, p-value < 2.2e-16
```

Assumptions fail, so I am using a permutation two-sample test.

```
kelpie.permTS=permTS(kelpie.visit$duration[kelpie.visit$octopus=="absent"],
                     kelpie.visit$duration[kelpie.visit$octopus=="present"],
       alternative="two.sided",method="exact.mc",control=permControl(nmc=10000))
kelpie.permTS
```

```
##
##  Exact Permutation Test Estimated by Monte Carlo
##
## data:  GROUP 1 and GROUP 2
## p-value = 2e-04
## alternative hypothesis: true mean GROUP 1 - mean GROUP 2 is not equal to 0
## sample estimates:
## mean GROUP 1 - mean GROUP 2
##                  -0.4268485
##
## p-value estimated from 10000 Monte Carlo replications
## 99 percent confidence interval on p-value:
##  0.000000000 0.001059383
```

Very significant!

Seeing what the mean difference in seconds is between kelp greenling visit duration when octopuses are absent versus octopuses are present.

```
aggregate(duration*60~octopus,data=kelpie.visit,FUN="mean")
```

```
##   octopus duration * 60
## 1  absent      10.85366
## 2 present      36.46457
```

5.3 Lingcod Analysis

5.3.1 Lingcod visitation frequency with octopuses absent vs. present.

In this chunk I am tallying Lingcod arrivals when octopuses are present versus absent.

```
lings=numeric()

for (i in 1:nrow(octos.present)){
  lings[i]=sum(events$Species=="lingcod"&
    events$Behavior=="frame_arrival"&
    events$`Deployment #`==octos.present$`Deployment #`[i]&
    events$realtime>octos.present$realtime[i]&
    events$realtime<octos.present$endtime[i])
}

lings.absent=sum(lings[octos.present$number==0])
lings.present=sum(lings[octos.present$number>0])
lings.absent
```

```
## [1] 138
```

```
lings.present
```

```
## [1] 76
```

Next I perform a chi square test to see if the lingcod visitation frequency is significantly different that random. Here, I am using the probability in each category (absent vs. present) as the proportion of time octopuses were absent versus present.

```
probs=c(min.absent/(min.absent+min.present),min.present/(min.absent+min.present))

chisq.test(c(lings.absent,lings.present),p=probs)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(lings.absent, lings.present)
## X-squared = 0.27315, df = 1, p-value = 0.6012
```

5.3.2 lingcod visitation duration when octopuses are absent versus present.

First, I make dataframes to hold my visit duration information and catalog potential bad arrivals and departure entries.

```
ling.visit=data.frame(deployment=as.numeric(NA),
           time=as.POSIXct(NA),
           duration=as.numeric(NA))

ling.badarrival=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))

ling.baddepar=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))
```

Next, I calculate the duration of each lingcod visit.

```
for (i in 1:length(deployments)){
  arrivals=events$realtime[events$Species=="lingcod"&
                     events$Behavior=="frame_arrival"&
                     events$`Deployment #`==deployments[i]]

  departures=events$realtime[events$Species=="lingcod"&
                     events$Behavior=="frame_departure"&
                     events$`Deployment #`==deployments[i]]

  if (length(arrivals)>0){
  depar.tags=c(rep("arrive",length(arrivals)),rep("depart",length(departures)))

  deploy.sub=data.frame(time=c(arrivals,departures),type=depar.tags)
  deploy.sub=deploy.sub[order(deploy.sub$time),]

      #Finding which arrivals are immediately followed by a departure
      visit.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
```

```
                          deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      # Now calculating the
      visit.dura=difftime(deploy.sub$time[visit.depar+1],deploy.sub$time[visit.depar],units = "min")
      visit.stub=data.frame(deployment=rep(deployments[i],length(visit.dura)),
                  time=deploy.sub$time[visit.depar],
                  duration=as.numeric(visit.dura))
      ling.visit=rbind(ling.visit,visit.stub)

      # Finding arrivals followed immediately by another arrival
      #(This should not be possible, so this is error locating)
      badarrival.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                               deploy.sub$type[2:nrow(deploy.sub)]=="arrive")
      badarrival.dura=difftime(deploy.sub$time[badarrival.depar+1],
                               deploy.sub$time[badarrival.depar],units = "min")
      badarrival.stub=data.frame(deployment=rep(deployments[i],length(badarrival.dura)),
                  time1=deploy.sub$time[badarrival.depar],
                  time2=deploy.sub$time[badarrival.depar+1],
                  duration=as.numeric(badarrival.dura))
      ling.badarrival=rbind(ling.badarrival,badarrival.stub)

      baddepar.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="depart"&
                             deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      baddepar.dura=difftime(deploy.sub$time[baddepar.depar+1],
                             deploy.sub$time[baddepar.depar],units = "min")
      baddepar.stub=data.frame(deployment=rep(deployments[i],length(baddepar.dura)),
                  time1=deploy.sub$time[baddepar.depar],
                  time2=deploy.sub$time[baddepar.depar+1],
                  duration=as.numeric(baddepar.dura))
      ling.baddepar=rbind(ling.baddepar,baddepar.stub)


    }
}


ling.visit=ling.visit[complete.cases(ling.visit),]
```

Adding if octopuses are present or absent during each visit to my dataframe

```
ling.visit$octopus="unknown"


for (i in 1:nrow(octos.present)){
  if (octos.present$number[i]>0){
    ling.visit$octopus[ling.visit$deployment==octos.present$`Deployment #`[i]&
                  ling.visit$time>octos.present$realtime[i]&
                  ling.visit$time<octos.present$endtime[i]]="present"
  }
  if (octos.present$number[i]==0){
    ling.visit$octopus[ling.visit$deployment==octos.present$`Deployment #`[i]&
                  ling.visit$time>octos.present$realtime[i]&
                  ling.visit$time<octos.present$endtime[i]]="absent"
  }
```

```
}

ling.visit=ling.visit[!ling.visit$octopus=="unknown",]
```

Adding if each ling visit was during the day or the night

```
ling.visit$daynight="night"
ling.visit$day_expected=0

for (i in 1:nrow(ling.visit)){
  sunrise=sunriset(bay, as.POSIXct(ling.visit$time[i]),
                   direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(ling.visit$time[i]),
                  direction="sunset", POSIXct.out=TRUE)$time
  if (ling.visit$time[i]>sunrise&ling.visit$time[i]<sunset) {
    ling.visit$daynight[i]="day"
  }
  ling.visit$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}
```

Getting distribution of observations for chi-squared test

```
ling.tab=
  as.vector(table(ling.visit$octopus))
```

Setting up probabilities for chi-squared test

```
ling.probs=
  c(min.absent/(min.absent+min.present),min.present/(min.absent+min.present))
```

Running chi-squared test

```
ling.chi=chisq.test(ling.tab,p=ling.probs)
ling.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  ling.tab
## X-squared = 0.27315, df = 1, p-value = 0.6012
```

Plotting chi-squared residuals

```
ling.res=matrix(ling.chi$residuals,ncol=2)
colnames(ling.res)=c("octopus absent","octopus present")
```

Testing my assumptions for T-Test

```
shapiro.test(ling.visit$duration[ling.visit$octopus=="present"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ling.visit$duration[ling.visit$octopus == "present"]
## W = 0.27692, p-value < 2.2e-16
```

```
shapiro.test(ling.visit$duration[ling.visit$octopus=="absent"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ling.visit$duration[ling.visit$octopus == "absent"]
## W = 0.36405, p-value < 2.2e-16
```

```
bartlett.test(ling.visit$duration,ling.visit$octopus)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  ling.visit$duration and ling.visit$octopus
## Bartlett's K-squared = 160.76, df = 1, p-value < 2.2e-16
```

Assumptions fail, so I am using a permutation two-sample test.

```
ling.permTS=permTS(ling.visit$duration[ling.visit$octopus=="absent"],
                   ling.visit$duration[ling.visit$octopus=="present"],
       alternative="two.sided",method="exact.mc",control=permControl(nmc=10000))
ling.permTS
```

```
##
##  Exact Permutation Test Estimated by Monte Carlo
##
## data:  GROUP 1 and GROUP 2
## p-value = 0.07099
## alternative hypothesis: true mean GROUP 1 - mean GROUP 2 is not equal to 0
## sample estimates:
## mean GROUP 1 - mean GROUP 2
##                  -0.1916317
##
## p-value estimated from 10000 Monte Carlo replications
## 99 percent confidence interval on p-value:
##  0.06162466 0.08087184
```

Not significant

Seeing what the mean difference is between lingcod visit duration when octopuses are absent versus octopuses are present.

```
aggregate(duration~octopus,data=ling.visit,FUN="max")
```

```
##   octopus duration
## 1  absent 2.366667
## 2 present 8.800000
```

5.4 Rockfish Analysis

5.4.1 rockfish visitation duration when octopuses are absent versus present.

First, I make dataframes to hold my visit duration information and catalog potential bad arrivals and departure entries.

```
rock.visit=data.frame(deployment=as.numeric(NA),
           time=as.POSIXct(NA),
           duration=as.numeric(NA))

rock.badarrival=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))

rock.baddepar=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))
```

Next, I calculate the duration of each rockfish visit.

```
for (i in 1:length(deployments)){
  arrivals=events$realtime[events$Species=="rockfish"&
                     events$Behavior=="frame_arrival"&
                     events$`Deployment #`==deployments[i]]

  departures=events$realtime[events$Species=="rockfish"&
                     events$Behavior=="frame_departure"&
                     events$`Deployment #`==deployments[i]]

  if (length(arrivals)>0){
  depar.tags=c(rep("arrive",length(arrivals)),rep("depart",length(departures)))

  deploy.sub=data.frame(time=c(arrivals,departures),type=depar.tags)
  deploy.sub=deploy.sub[order(deploy.sub$time),]

      #Finding which arrivals are immediately followed by a departure
      visit.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                          deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      # Now calculating the
      visit.dura=difftime(deploy.sub$time[visit.depar+1],
                          deploy.sub$time[visit.depar],units = "min")
      visit.stub=data.frame(deployment=rep(deployments[i],length(visit.dura)),
                 time=deploy.sub$time[visit.depar],
                 duration=as.numeric(visit.dura))
      rock.visit=rbind(rock.visit,visit.stub)

      # Finding arrivals followed immediately by another arrival
      #(This should not be possible, so this is error locating)
      badarrival.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                               deploy.sub$type[2:nrow(deploy.sub)]=="arrive")
      badarrival.dura=difftime(deploy.sub$time[badarrival.depar+1],
```

```
                                deploy.sub$time[badarrival.depar],units = "min")
      badarrival.stub=data.frame(deployment=rep(deployments[i],length(badarrival.dura)),
                  time1=deploy.sub$time[badarrival.depar],
                  time2=deploy.sub$time[badarrival.depar+1],
                  duration=as.numeric(badarrival.dura))
      rock.badarrival=rbind(rock.badarrival,badarrival.stub)

      baddepar.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="depart"&
                             deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      baddepar.dura=difftime(deploy.sub$time[baddepar.depar+1],
                             deploy.sub$time[baddepar.depar],units = "min")
      baddepar.stub=data.frame(deployment=rep(deployments[i],length(baddepar.dura)),
                  time1=deploy.sub$time[baddepar.depar],
                  time2=deploy.sub$time[baddepar.depar+1],
                  duration=as.numeric(baddepar.dura))
      rock.baddepar=rbind(rock.baddepar,baddepar.stub)


    }
}


rock.visit=rock.visit[complete.cases(rock.visit),]
```

Adding if octopuses are present or absent during each visit to my dataframe

```
rock.visit$octopus="unknown"


for (i in 1:nrow(octos.present)){
  if (octos.present$number[i]>0){
    rock.visit$octopus[rock.visit$deployment==octos.present$`Deployment #`[i]&
                rock.visit$time>octos.present$realtime[i]&
                rock.visit$time<octos.present$endtime[i]]="present"
  }
  if (octos.present$number[i]==0){
    rock.visit$octopus[rock.visit$deployment==octos.present$`Deployment #`[i]&
                rock.visit$time>octos.present$realtime[i]&
                rock.visit$time<octos.present$endtime[i]]="absent"
  }
}

rock.visit=rock.visit[!rock.visit$octopus=="unknown",]
```

Adding if each rock visit was during the day or the night

```
rock.visit$daynight="night"
rock.visit$day_expected=0

for (i in 1:nrow(rock.visit)){
  sunrise=sunriset(bay, as.POSIXct(rock.visit$time[i]),
                   direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(rock.visit$time[i]),
```

```
                  direction="sunset", POSIXct.out=TRUE)$time
  if (rock.visit$time[i]>sunrise&rock.visit$time[i]<sunset) {
    rock.visit$daynight[i]="day"
  }
  rock.visit$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}
```

Getting distribution of observations for chi-squared test

```
rock.tab=
  as.vector(table(rock.visit$octopus))
```

Setting up probabilities for chi-squared test

```
rock.probs=
  c(min.absent/(min.absent+min.present),min.present/(min.absent+min.present))
```

Running chi-squared test

```
rock.chi=chisq.test(rock.tab,p=rock.probs)
rock.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  rock.tab
## X-squared = 39.238, df = 1, p-value = 3.752e-10
```

Plotting chi-squared residuals

```
rock.res=matrix(rock.chi$residuals,ncol=2)
colnames(rock.res)=c("octopus absent","octopus present")
```

Testing my assumptions for T-Test

```
shapiro.test(rock.visit$duration[rock.visit$octopus=="present"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rock.visit$duration[rock.visit$octopus == "present"]
## W = 0.4087, p-value < 2.2e-16
```

```
shapiro.test(rock.visit$duration[rock.visit$octopus=="absent"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rock.visit$duration[rock.visit$octopus == "absent"]
## W = 0.36315, p-value < 2.2e-16
```

```
bartlett.test(rock.visit$duration,rock.visit$octopus)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  rock.visit$duration and rock.visit$octopus
## Bartlett's K-squared = 0.23026, df = 1, p-value = 0.6313
```

Assumptions fail, so I am using a permutation two-sample test.

```
rock.permTS=permTS(rock.visit$duration[rock.visit$octopus=="absent"],
                   rock.visit$duration[rock.visit$octopus=="present"],
       alternative="two.sided",method="exact.mc",control=permControl(nmc=10000))
rock.permTS
```

```
##
##  Exact Permutation Test Estimated by Monte Carlo
##
## data:  GROUP 1 and GROUP 2
## p-value = 0.6129
## alternative hypothesis: true mean GROUP 1 - mean GROUP 2 is not equal to 0
## sample estimates:
## mean GROUP 1 - mean GROUP 2
##                 -0.02992012
##
## p-value estimated from 10000 Monte Carlo replications
## 99 percent confidence interval on p-value:
##  0.5891404 0.6368301
```

Not Significant

Seeing what the mean difference is between Rockfish visit duration when octopuses are absent versus octopuses are present.

```
aggregate(duration~octopus,data=rock.visit,FUN="mean")
```

```
##   octopus  duration
## 1  absent 0.2311910
## 2 present 0.2611111
```

5.5 Red Rock Crab Analysis

5.5.1 crab visitation duration when octopuses are absent versus present.

First, I make dataframes to hold my visit duration information and catalog potential bad arrivals and departure entries.

```
crab.visit=data.frame(deployment=as.numeric(NA),
           time=as.POSIXct(NA),
           duration=as.numeric(NA))
```

```
crab.badarrival=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))

crab.baddepar=data.frame(deployment=as.numeric(NA),
           time1=as.POSIXct(NA),
           time2=as.POSIXct(NA),
           duration=as.numeric(NA))
```

Next, I calculate the duration of each crab visit.

```
for (i in 1:length(deployments)){
  arrivals=events$realtime[events$Species=="red rock crab"&
                    events$Behavior=="frame_arrival"&
                    events$`Deployment #`==deployments[i]]

  departures=events$realtime[events$Species=="red rock crab"&
                    events$Behavior=="frame_departure"&
                    events$`Deployment #`==deployments[i]]

  if (length(arrivals)>0){
  depar.tags=c(rep("arrive",length(arrivals)),rep("depart",length(departures)))

  deploy.sub=data.frame(time=c(arrivals,departures),type=depar.tags)
  deploy.sub=deploy.sub[order(deploy.sub$time),]

      #Finding which arrivals are immediately followed by a departure
      visit.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                          deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      # Now calculating the
      visit.dura=difftime(deploy.sub$time[visit.depar+1],
                          deploy.sub$time[visit.depar],units = "min")
      visit.stub=data.frame(deployment=rep(deployments[i],length(visit.dura)),
                 time=deploy.sub$time[visit.depar],
                 duration=as.numeric(visit.dura))
      crab.visit=rbind(crab.visit,visit.stub)

      # Finding arrivals followed immediately by another arrival
      #(This should not be possible, so this is error locating)
      badarrival.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="arrive"&
                               deploy.sub$type[2:nrow(deploy.sub)]=="arrive")
      badarrival.dura=difftime(deploy.sub$time[badarrival.depar+1],
                               deploy.sub$time[badarrival.depar],units = "min")
      badarrival.stub=data.frame(deployment=rep(deployments[i],length(badarrival.dura)),
                 time1=deploy.sub$time[badarrival.depar],
                 time2=deploy.sub$time[badarrival.depar+1],
                 duration=as.numeric(badarrival.dura))
      crab.badarrival=rbind(crab.badarrival,badarrival.stub)

      baddepar.depar=which(deploy.sub$type[1:(nrow(deploy.sub)-1)]=="depart"&
                             deploy.sub$type[2:nrow(deploy.sub)]=="depart")
      baddepar.dura=difftime(deploy.sub$time[baddepar.depar+1],
```

```
                              deploy.sub$time[baddepar.depar],units = "min")
      baddepar.stub=data.frame(deployment=rep(deployments[i],length(baddepar.dura)),
                  time1=deploy.sub$time[baddepar.depar],
                  time2=deploy.sub$time[baddepar.depar+1],
                  duration=as.numeric(baddepar.dura))
      crab.baddepar=rbind(crab.baddepar,baddepar.stub)


    }
}


crab.visit=crab.visit[complete.cases(crab.visit),]
```

Adding if octopuses are present or absent during each visit to my dataframe

```
crab.visit$octopus="unknown"


for (i in 1:nrow(octos.present)){
  if (octos.present$number[i]>0){
    crab.visit$octopus[crab.visit$deployment==octos.present$`Deployment #`[i]&
                  crab.visit$time>octos.present$realtime[i]&
                  crab.visit$time<octos.present$endtime[i]]="present"
  }
  if (octos.present$number[i]==0){
    crab.visit$octopus[crab.visit$deployment==octos.present$`Deployment #`[i]&
                  crab.visit$time>octos.present$realtime[i]&
                  crab.visit$time<octos.present$endtime[i]]="absent"
  }
}

crab.visit=crab.visit[!crab.visit$octopus=="unknown",]
```

Adding if each crab visit was during the day or the night

```
crab.visit$daynight="night"
crab.visit$day_expected=0

for (i in 1:nrow(crab.visit)){
  sunrise=sunriset(bay, as.POSIXct(crab.visit$time[i]),
                   direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(crab.visit$time[i]),
                  direction="sunset", POSIXct.out=TRUE)$time
  if (crab.visit$time[i]>sunrise&crab.visit$time[i]<sunset) {
    crab.visit$daynight[i]="day"
  }
  crab.visit$day_expected[i]=as.numeric(difftime(sunset,sunrise,units="hours"))/24
}
```

Getting distribution of observations for chi-squared test

```
crab.tab=
  as.vector(table(crab.visit$octopus))
```

Setting up probabilities for chi-squared test

```
crab.probs=
  c(min.absent/(min.absent+min.present),min.present/(min.absent+min.present))
```

Running chi-squared test

```
crab.chi=chisq.test(crab.tab,p=crab.probs)
crab.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  crab.tab
## X-squared = 14.281, df = 1, p-value = 0.0001574
```

Plotting chi-squared residuals

```
crab.res=matrix(crab.chi$residuals,ncol=2)
colnames(crab.res)=c("octopus absent","octopus present")
```

Testing my assumptions for T-Test

```
shapiro.test(crab.visit$duration[crab.visit$octopus=="present"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  crab.visit$duration[crab.visit$octopus == "present"]
## W = 0.35578, p-value < 2.2e-16
```

```
shapiro.test(crab.visit$duration[crab.visit$octopus=="absent"])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  crab.visit$duration[crab.visit$octopus == "absent"]
## W = 0.3447, p-value < 2.2e-16
```

```
bartlett.test(crab.visit$duration,crab.visit$octopus)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  crab.visit$duration and crab.visit$octopus
## Bartlett's K-squared = 1.1881, df = 1, p-value = 0.2757
```

Assumptions fail, so I am using a permutation two-sample test.

```
crab.permTS=permTS(crab.visit$duration[crab.visit$octopus=="absent"],
                   crab.visit$duration[crab.visit$octopus=="present"],
       alternative="two.sided",method="exact.mc",control=permControl(nmc=10000))
```

Not significant!

Seeing what the mean difference is between lingcod visit duration when octopuses are absent versus octopuses are present.

```
aggregate(duration~octopus,data=crab.visit,FUN="mean")
```

```
##   octopus duration
## 1  absent 2.272619
## 2 present 2.586275
```

6 Plotting intraspecific and conspecific octopus interactions

We first append together all of the Pearson residuals from frequency chi square tests

```
all.res=
  cbind(
    kelpie.res,
    ling.res,
    rock.res,
    crab.res,
    octo.res
  )
```

Then, we append together all of the p-values.

```
all.chi.p=
  cbind(
    kelpie.chi$p.value,
    ling.chi$p.value,
    rock.chi$p.value,
    crab.chi$p.value,
    octo.chi$p.value
  )
```

```
colnames(all.res)=rep("",ncol(all.res))
rownames(all.res)=rep("",nrow(all.res))
```

Finally, we plot the data into a large boxplot.

```
svg(filename="Figure3.svg",width=9,height=7)
plot(c(0,1),c(0,1),type="n",axes=F,ylab="",xlab="")


par(fig=c(0.076,0.957,0,0.5),new=T)
```

```
corrplot(all.res,is.cor=F,col.lim=c(-5,5),tl.col="black",
         addCoef.col = "black",cl.pos=F,addgrid.col=NA)
for (i in c(0,2,4,6,8)){
  lines(c(.5+i,.5+i,2.5+i,2.5+i,.5+i),c(.5,1.9,1.9,.5,.5),lwd=3)
}

for (i in 0:4){
  text(2*i+1.15,0.4,"absent",srt=90,pos=2)
  text(2*i+2.15,0.4,"present",srt=90,pos=2)
}

for (i in 0:4){
  if (all.chi.p[i+1]<=0.05){
    lines(c((i*2)+1,(i*2)+2),c(1.5,1.5),lwd=2)
    text((i*2)+1.5,1.7,paste("p=",format(signif(all.chi.p[i+1],2),scientific=F),sep=""))
  }
}


start=0.001
box.width=0.310
box.next=0.171
lower.box.bottom=0.21
lower.box.top=0.8
upper.box.bottom=0.536
upper.box.top=1
bottom.max=0.7
top.min=0.82
top.max=100
top.at=c(1)
top.log="y"

par(fig=c(start,start+box.width,lower.box.bottom,lower.box.top),new=T)
boxplot(duration~octopus,data=kelpie.visit,range=0,ylim=c(0,bottom.max),axes=F,
        ylab="",xlab="")
axis(2,at=c(0,0.3,0.6),las=1)
box(lwd=3)
text(1,mean(kelpie.visit$duration[kelpie.visit$octopus=="absent"]),"+",cex=2)
text(2,mean(kelpie.visit$duration[kelpie.visit$octopus=="present"]),"+",cex=2)

par(fig=c(start,start+box.width,upper.box.bottom,upper.box.top),new=T)
boxplot(duration~octopus,data=kelpie.visit,range=0,ylim=c(top.min,top.max),axes=F,
        ylab="",xlab="",log=top.log)
axis(2,at=c(1,10,100),labels=c("1","10","100"),las=1)
box(lwd=3)
if (kelpie.permTS$p.value<=0.05){
  lines(c(1,2),c(exp(log(max(kelpie.visit$duration))+0.5),
                 exp(log(max(kelpie.visit$duration))+0.5)),lwd=2)
  text(1.5,exp(log(max(kelpie.visit$duration))+1),
       paste("p=",format(signif(kelpie.permTS$p.value,2),scientific = F),sep=""))

}
```

```
par(fig=c(start+box.next*1,start+box.next*1+box.width,lower.box.bottom,lower.box.top),new=T)
boxplot(duration~octopus,data=ling.visit,range=0,ylim=c(0,bottom.max),axes=F,ylab="",xlab="")
box(lwd=3)
text(1,mean(ling.visit$duration[ling.visit$octopus=="absent"]),"+",cex=2)
text(2,mean(ling.visit$duration[ling.visit$octopus=="present"]),"+",cex=2)

par(fig=c(start+box.next*1,start+box.next*1+box.width,upper.box.bottom,upper.box.top),new=T)
boxplot(duration~octopus,data=ling.visit,range=0,ylim=c(top.min,top.max),axes=F,ylab="",
        xlab="",log=top.log)
box(lwd=3)
if (ling.permTS$p.value<=0.05){
  lines(c(1,2),c(exp(log(max(ling.visit$duration))+0.5),
                 exp(log(max(ling.visit$duration))+0.5)),lwd=2)
  text(1.5,exp(log(max(ling.visit$duration))+1),
       paste("p=",format(signif(ling.permTS$p.value,2),scientific=F),sep=""))
}


par(fig=c(start+box.next*2,start+box.next*2+box.width,lower.box.bottom,lower.box.top),new=T)
boxplot(duration~octopus,data=rock.visit,range=0,ylim=c(0,bottom.max),axes=F,
        ylab="",xlab="")
box(lwd=3)
text(1,mean(rock.visit$duration[rock.visit$octopus=="absent"]),"+",cex=2)
text(2,mean(rock.visit$duration[rock.visit$octopus=="present"]),"+",cex=2)

par(fig=c(start+box.next*2,start+box.next*2+box.width,upper.box.bottom,upper.box.top),new=T)
boxplot(duration~octopus,data=rock.visit,range=0,ylim=c(top.min,top.max),axes=F,ylab="",
        xlab="",log=top.log)
box(lwd=3)
if (rock.permTS$p.value<=0.05){
  lines(c(1,2),c(exp(log(max(rock.visit$duration))+0.5),
                 exp(log(max(rock.visit$duration))+0.5)),lwd=2)
  text(1.5,exp(log(max(rock.visit$duration))+1),
       paste("p=",format(signif(rock.permTS$p.value,2),scientific=F),sep=""))
}


par(fig=c(start+box.next*3,start+box.next*3+box.width,lower.box.bottom,lower.box.top),new=T)
boxplot(duration~octopus,data=crab.visit,range=0,ylim=c(0,bottom.max),axes=F,ylab="",xlab="")
box(lwd=3)

par(fig=c(start+box.next*3,start+box.next*3+box.width,upper.box.bottom,upper.box.top),new=T)
boxplot(duration~octopus,data=crab.visit,range=0,ylim=c(top.min,top.max),axes=F,ylab="",
        xlab="",log=top.log)
box(lwd=3)
if (crab.permTS$p.value<=0.05){
  lines(c(1,2),c(exp(log(max(crab.visit$duration))+0.5),
                 exp(log(max(crab.visit$duration))+0.5)),lwd=2)
  text(1.5,exp(log(max(crab.visit$duration))+1),
       paste("p=",format(signif(crab.permTS$p.value,2),scientific=F),sep=""))
}
text(1,mean(crab.visit$duration[crab.visit$octopus=="absent"]),"+",cex=2)
text(2,mean(crab.visit$duration[crab.visit$octopus=="present"]),"+",cex=2)
```

```
par(fig=c(start+box.next*4,start+box.next*4+box.width,lower.box.bottom,lower.box.top),new=T)
boxplot(duration~resident,data=octo.visit,range=0,ylim=c(0,bottom.max),axes=F,
        ylab="",xlab="")
box(lwd=3)

par(fig=c(start+box.next*4,start+box.next*4+box.width,upper.box.bottom,upper.box.top),new=T)
boxplot(duration~resident,data=octo.visit,range=0,ylim=c(top.min,top.max),axes=F,ylab="",
        xlab="",log=top.log)
box(lwd=3)
if (octo.permTS$p.value<=0.05){
  lines(c(1,2),c(exp(log(max(octo.visit$duration))+0.5),
                 exp(log(max(octo.visit$duration))+0.5)),lwd=2)
  text(1.5,exp(log(max(octo.visit$duration))+1),
       paste("p=",format(signif(octo.permTS$p.value,2),scientific=F),sep=""))
}
text(1,mean(octo.visit$duration[octo.visit$resident=="absent"]),"+",cex=2)
text(2,mean(octo.visit$duration[octo.visit$resident=="present"]),"+",cex=2)


species.line1=1
species.line2=2
n.line=0

par(fig=c(0,1,0,1),new=T)
plot(c(0,1),c(0,1),type="n",axes=F,ylab="",xlab="")
mtext("Visit duration (min)",side=2,line=2.5,cex=1.5,adj=0.7)
mtext(paste("n=",nrow(kelpie.visit),sep=""),side=3,line=n.line,cex=1,adj=0.08)
mtext("decagrammus",side=3,line=species.line1,cex=1,adj=0.05,font=3)
mtext("Hexagrammos",side=3,line=species.line2,cex=1,adj=0.05,font=3)
mtext(paste("n=",nrow(ling.visit),sep=""),side=3,line=n.line,cex=1,adj=0.28)
mtext("elongatus",side=3,line=species.line1,cex=1,adj=0.27,font=3)
mtext("Ophiodon",side=3,line=species.line2,cex=1,adj=0.27,font=3)
mtext(paste("n=",nrow(rock.visit),sep=""),side=3,line=n.line,cex=1,adj=0.5)
mtext("Sebastes sp.",side=3,line=species.line1,cex=1,adj=0.5,font=3)
mtext(paste("n=",nrow(crab.visit),sep=""),side=3,line=n.line,cex=1,adj=0.71)
mtext("Cancer productus",side=3,line=species.line1,cex=1,adj=0.73,font=3)
mtext(paste("n=",nrow(octo.visit),sep=""),side=3,line=n.line,cex=1,adj=0.92)
mtext("Octopus rubescens",side=3,line=species.line1,cex=1,adj=0.99,font=3)
mtext("non-resident",side=3,line=species.line2,cex=1,adj=0.95,font=3)
mtext("octopus present or absent",side=1,line=3.5,cex=1.5)
dev.off()
```

```
## pdf
##   2
```

Next, we convert the image to a png so that it can be inserted into the RMarkdown

```
cairosvg Figure3.svg -o Figure3.png -d 300
```

Then, we convert it to an eps for publication.

```
inkscape Figure3.svg -o Figure3.eps --export-ignore-filters --export-ps-level=3
```

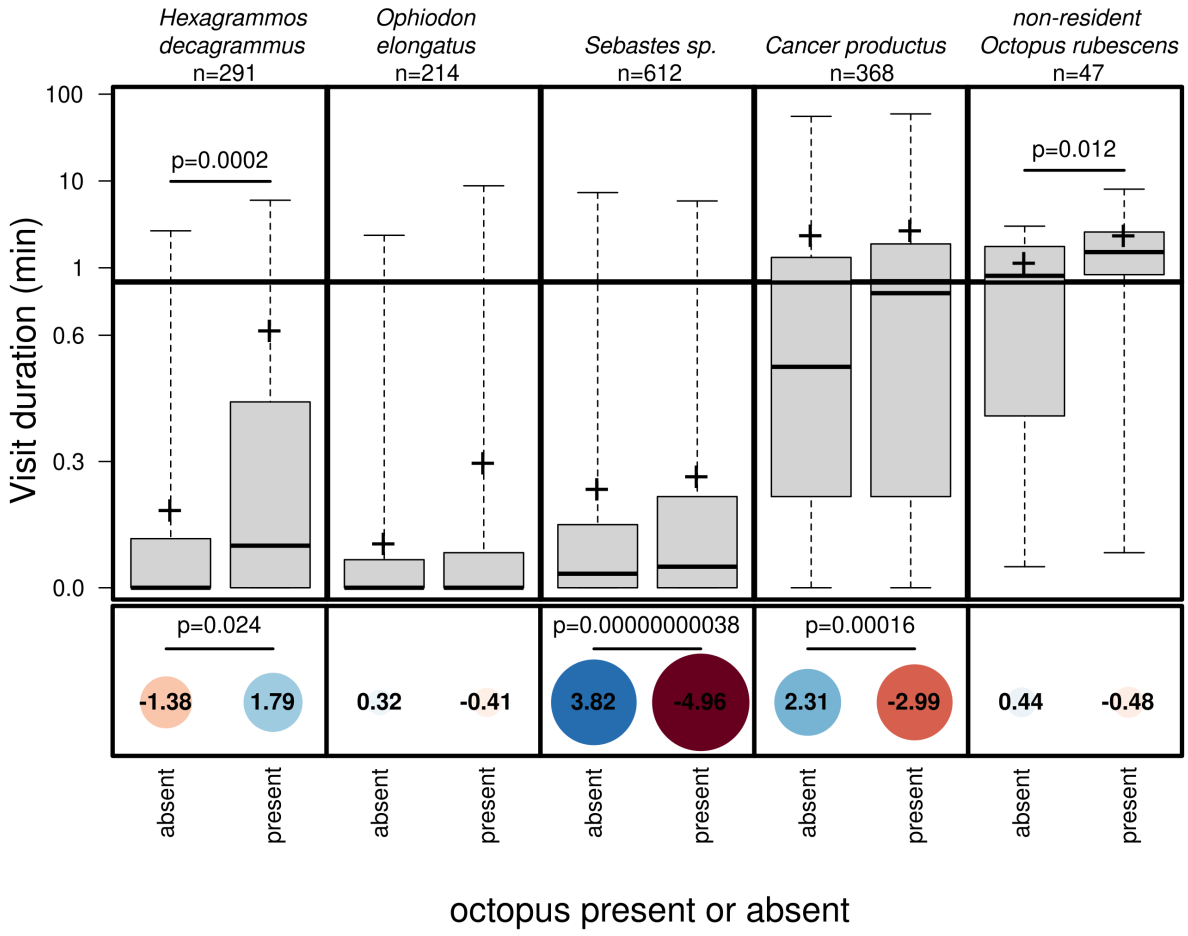

Figure 2: Commonly encountered species were evaluated for visitation duration and frequency, with and without a resident octopus present. Four non-octopus species were examined with a fifth comparison of non- resident octopus visitations. Top panel displays boxplot of visitation durations for each species when an octopus was not present or present in the camera field of view. Plusses (+) indicate average visitation durations. Y-axis is broken at 0.7 mins, above which is logarithmic in order to display long tails. Bottom panel displays Pearson residuals of chi-square analysis of visitation frequency when octopuses were not present or present in the camera field of view.

7 Are you more likely to encounter an octopus in the frame at a random time at night or during the day?

First, make a list of every minute from every duration from the octos.present object,

```
times=seq(from=octos.present$realtime[1],to=octos.present$endtime[1],by="1 min")
octos=rep(octos.present$number[1],length(times))

for (i in 2:nrow(octos.present)){
```

```
  times=c(times,seq(from=octos.present$realtime[i],to=octos.present$endtime[i],by="1 min"))
  octos=c(octos,rep(octos.present$number[i],length(seq(from=octos.present$realtime[i],to=octos.present$
}
```

Next, for each of those times, I will determine if it is during day or night.

```
daynight=rep("night",length(times))

dates=unique(as.Date(times))

for (i in 1:length(dates)){
  sunrise=sunriset(bay, as.POSIXct(dates[i]), direction="sunrise", POSIXct.out=TRUE)$time
  sunset=sunriset(bay,  as.POSIXct(dates[i]), direction="sunset", POSIXct.out=TRUE)$time
  daynight[times>sunrise&times<sunset]="day"
}
```

Sampling 612 night times and 612 day times to see if an octopus is present, and then comparing if the number of samples with an octopus is present is different between the two samples. Essentially what we are trying to look at here: If you were to visit the octopus dens randomly 612 times during the night, or visit the octopus dens 612 times during the day, would you be more likely to encounter and octopus during the day or night?

```
daynight.samp=c(sum(sample(octos[daynight=="night"]>0,612)),sum(sample(octos[daynight=="day"]>0,612)))
daynight.samp.chi=chisq.test(daynight.samp)
daynight.samp.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  daynight.samp
## X-squared = 0.43946, df = 1, p-value = 0.5074
```

The frequency of octopus encounters is not significantly different between day and night. It appears the answer is no, you would not be more likely to encounter an octopus on a random visit to the bottle dens in the day or night.

8 Frequency of periscoping and fortifying when 1 octopus is present versus multiple

These two behaviors by octopuses seem like they could be influenced by the presence or absence of other octopuses. In this analysis we look to see if they happen more or less frequently when the octopus doing the behavior is alone in the camera field of view, or if it is not alone. First, we get the relative amounts of time when there is only one octopus present versus multiple octopuses present. In this calculation, we multiply the minutes that multiple how many octopuses are present to account for the multiple octopuses present and the multiplication of the probability that you would observe a specific octopus behavior in that time (for instance: you are twice as likely to observe any specific octopus behavior when two octopuses are present).

```
octo.one=sum(octos.present$duration[octos.present$number==1])
octo.more=sum(octos.present$duration[octos.present$number>1]*
                octos.present$number[octos.present$number>1])
octo.one
```

```
## [1] 9436.317
```

```
octo.more
```

```
## [1] 13119.7
```

8.1 Probabilities of 1 versus >1 octopus present

```
perifort.probs=c(octo.one/(octo.one+octo.more),octo.more/(octo.one+octo.more))
perifort.probs
```

```
## [1] 0.4183503 0.5816497
```

8.2 Finding number of fortify events with 1 octopus present versus more than 1

```
fort=numeric()

for (i in 1:nrow(octos.present)){
  fort[i]=sum(events$Behavior=="fortify"&
    events$`Deployment #`==octos.present$`Deployment #`[i]&
    events$realtime>octos.present$realtime[i]&
    events$realtime<octos.present$endtime[i])
}

fort.none=sum(fort[octos.present$number==0]) #negative control... kinda...
fort.one=sum(fort[octos.present$number==1])
fort.more=sum(fort[octos.present$number>1])
fort.none
```

```
## [1] 0
```

```
fort.one
```

```
## [1] 68
```

```
fort.more
```

```
## [1] 82
```

```
fort.multi.chi=chisq.test(c(fort.one,fort.more),p=perifort.probs)
fort.multi.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(fort.one, fort.more)
## X-squared = 0.7544, df = 1, p-value = 0.3851
```

It appears

8.3 Finding number of periscope events with 1 octopus present versus more than 1

```
peri=numeric()

for (i in 1:nrow(octos.present)){
  peri[i]=sum(events$Behavior=="periscope"&
    events$Species=="octopus_rubescens"&
    events$`Deployment #`==octos.present$`Deployment #`[i]&
    events$realtime>octos.present$realtime[i]&
    events$realtime<octos.present$endtime[i])
}

peri.none=sum(peri[octos.present$number==0]) #negative control... kinda...
peri.one=sum(peri[octos.present$number==1])
peri.more=sum(peri[octos.present$number>1])
peri.none
```

```
## [1] 0
```

```
peri.one
```

```
## [1] 31
```

```
peri.more
```

```
## [1] 12
```

```
peri.multi.chi=chisq.test(c(peri.one,peri.more),p=perifort.probs)
peri.multi.chi
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(peri.one, peri.more)
## X-squared = 16.179, df = 1, p-value = 5.763e-05
```

8.4 Chi square residuals plots

Now, making a plot of all of those final chi-squared analyses.

```
daynight.samp.plot=t(as.matrix(daynight.samp.chi$residuals))
colnames(daynight.samp.plot)=c("","")
rownames(daynight.samp.plot)=""


perifort.plot=as.matrix(rbind(fort.multi.chi$residuals,
peri.multi.chi$residuals))
colnames(perifort.plot)=c("","")
rownames(perifort.plot)=c("","")
```

```
testcorr=rbind(daynight.samp.plot,c(NA,NA),perifort.plot,c(NA,NA),reach.res)
```

```
svg(filename="Figure4.svg",width=7,height=15)
par(fig=c(0,1,0.1,0.95))
corrplot(testcorr,is.cor=F,tl.col="black",
         addCoef.col="black",addgrid.col=NA,col.lim=c(-5,5),
#          cl.cex=1.5,cl.ratio=0.3
          cl.pos="n"
         )
for (i in c(0,2,3,5)){
  lines(c(.5,.5,2.5,2.5,.5),c(.5+i,1.5+i,1.5+i,.5+i,.5+i),lwd=3)
}


axis(1,at=c(1,2),lwd=0,labels = c("present","absent"),cex.axis=2,padj=3.5)
points(c(1,2,1,2),c(2,2,5,5),pch=21,bg="white",col="white",cex=3)
# This code above is a real janky way to hide the questions marks


text(c(1,2),c(5.35,5.35),c("day","night"),cex=2)
text(c(1,2),c(2.35,2.35),c("1",">1"),cex=2)
text(c(1.5),c(2.1),c("octopuses in frame"),cex=2)
mtext("resident octopus",side=1,cex=2,line=8)
text(0.4,4,"fortify",srt=90,cex=2)
text(0.4,3,"periscope",srt=90,cex=2)

if (reach.chi$p.value>=0.001){
  text(1.5,1.4,paste0("p=",signif(reach.chi$p.value,digits=2)),cex=1.5)
} else (text(1.5,1.4,"p<0.001",cex=1.5))

if (peri.multi.chi$p.value>=0.001){
  text(1.5,3.4,paste0("p=",signif(peri.multi.chi$p.value,digits=2)),cex=1.5)
} else (text(1.5,3.4,"p<0.001",cex=1.5))
```

```
## NULL
```

```
if (fort.multi.chi$p.value>=0.001){
  text(1.5,4.4,paste0("p=",signif(fort.multi.chi$p.value,digits=2)),cex=1.5)
} else (text(1.5,4.4,"p<0.001",cex=1.5))
if (daynight.samp.chi$p.value>=0.001){
  mtext(paste0("p=",signif(daynight.samp.chi$p.value,digits=2)),side=3,line=1.7,cex=1.5)
} else (mtext("p<0.001",side=3,line=1.7,cex=1.5))

mtext("A",side=3,line=3.2,cex=4,adj=0.2)
text(0.53,4.64,"B",cex=4)
text(0.53,1.64,"C",cex=4)


#mtext("Pearson residuals",side=4,cex=2,line=-3)
#text(0.2,6,"A",cex=3)


dev.off()
```

```
## pdf
##   2
```

```
cairosvg Figure4.svg -o Figure4.png -d 300
```

```
inkscape Figure4.svg -o Figure4.eps --export-ignore-filters --export-ps-level=3
```

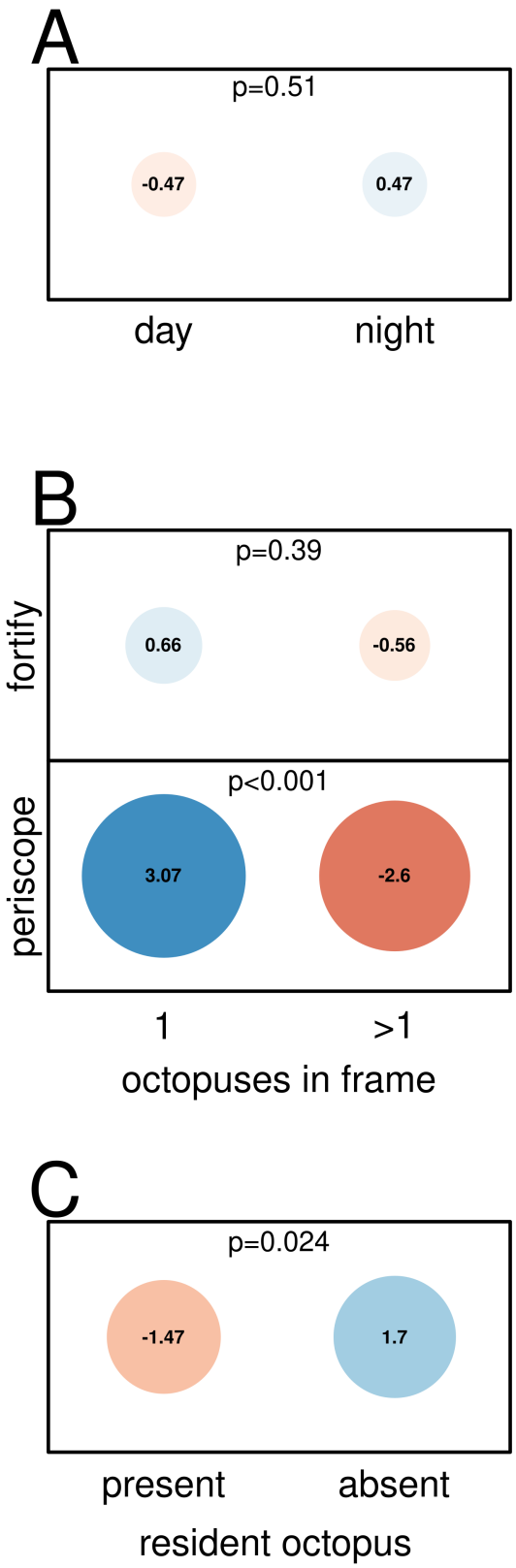

Figure 3: Pearson residuals for various chi-square analyses. (A) Relationship of octopus presence during day vs. night, checking for potential bias in frequency of octopus presence. (B) Influence of conspecific presence on octopus behaviors (# of octopuses within the camera frame 1 vs. more than 1); fortification and periscoping. (C) Frequency of non-resident (visiting) octopus interacting with bottle interior with and without a resident octopus present.