Masters

Emily Burnett

23/06/2021

Thesis R Markdown Code

Installing packages for data manipulation

```
require("readxl")
```

```
## Loading required package: readxl
```

```
## Warning: package 'readxl' was built under R version 4.0.5
```

```
require(c("RSQLite", "lubridate"))
```

```
## Warning in if (!loaded) {: the condition has length > 1 and only the first
## element will be used
```

```
## c("Loading required package: c", "Loading required package: RSQLite", "Loading required package: lubr
```

```
require(lubridate)
```

```
## Loading required package: lubridate
```

```
## Warning: package 'lubridate' was built under R version 4.0.5
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
require(RSQLite)
```

```
## Loading required package: RSQLite
```

```
## Warning: package 'RSQLite' was built under R version 4.0.5
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Importing databases into R

```
#Implementing 24th of June SQlite database
SQLite.database5 <-"D:/MScWork/PAMGuard/PAMDatabase/Good_Data/24062021data-no_clicks1.sqlite3"
database.con5 <- dbConnect(SQLite(), SQLite.database5)
dbListTables(database.con5)

#Implementing 15th of June SQlite database
SQLite.database4 <-"D:/MScWork/PAMGuard/PAMDatabase/Good_Data/15062021data-no_clicks.sqlite3"
database.con4 <- dbConnect(SQLite(), SQLite.database4)
dbListTables(database.con4)

#Implementing 9th of June SQlite database
SQLite.database3 <-"D:/MScWork/PAMGuard/PAMDatabase/Good_Data/09062021data-no_clicks.sqlite3"
database.con3 <- dbConnect(SQLite(), SQLite.database3)
dbListTables(database.con3)

#Implementing 8th of June SQlite database
SQLite.database2 <-"D:/MScWork/PAMGuard/PAMDatabase/Good_Data/08062021data2-no_clicks.sqlite3"
database.con2 <- dbConnect(SQLite(), SQLite.database2)
dbListTables(database.con2)

#Implementing 7th of June SQlite database
SQLite.database1 <-"D:/MScWork/PAMGuard/PAMDatabase/Good_Data/07062021data2-no_clicks.sqlite3"
database.con1 <- dbConnect(SQLite(), SQLite.database1)
dbListTables(database.con1)
```

Cutting databases down to Whistle moan detector

```
###24th
whistles5 <- dbReadTable(database.con5, "Whistle_and_Moan_Detector")
summary(whistles5)
str(whistles5)
```

```
###15th
whistles4 <- dbReadTable(database.con4, "Whistle_and_Moan_Detector")
summary(whistles4)
str(whistles4)

###9th
whistles3 <- dbReadTable(database.con3, "Whistle_and_Moan_Detector")
summary(whistles3)
str(whistles3)

###8th
whistles2 <- dbReadTable(database.con2, "Whistle_and_Moan_Detector")
summary(whistles2)
str(whistles2)

###7th
whistles1 <- dbReadTable(database.con1, "Whistle_and_Moan_Detector")
summary(whistles1)
str(whistles1)
```

Combining data sets into one database

```
#(only used first 7th,8th,9th for more streamlined analysis)
data<- rbind(whistles1, whistles2, whistles3)
str(data)
```

Filtering false detections and unnecessary frequencies from PAMGuard data

```
##Making sure only noise >3kHz (fundamental frequency) was used

data.high.freq <- data [which(data$lowFreq>=3200),]
data <- data.high.freq

##Removing false whistles (boat noise detected as whistles) by PAMGuard

data$freq.diff <- abs(data$lowFreq - data$highFreq)
data.no.noise <- data[which(data$freq.diff>500),]
data <- data.no.noise
str(data)
```

Making sure time was in seconds rather than miliseconds for easier analysis

```
##Time with milliseconds

data$UTC[1]
data$duration[1]
```

```
data$posix <- as.POSIXct(data$UTC, format="%Y-%m-%d %H:%M:%S", tz="UTC")

##Time in Posix in seconds
data$posix[1]
```

Removing boat noise by filtering low frequency differences out

```
quantile(data$freq.diff)
data1<-data[ data$freq.diff > quantile(data$freq.diff , 0.25 ) , ]
data1$posix <- as.POSIXct(data1$UTC, format="%Y-%m-%d %H:%M:%S", tz="UTC")
```

Formatting data to make 1 minute metric

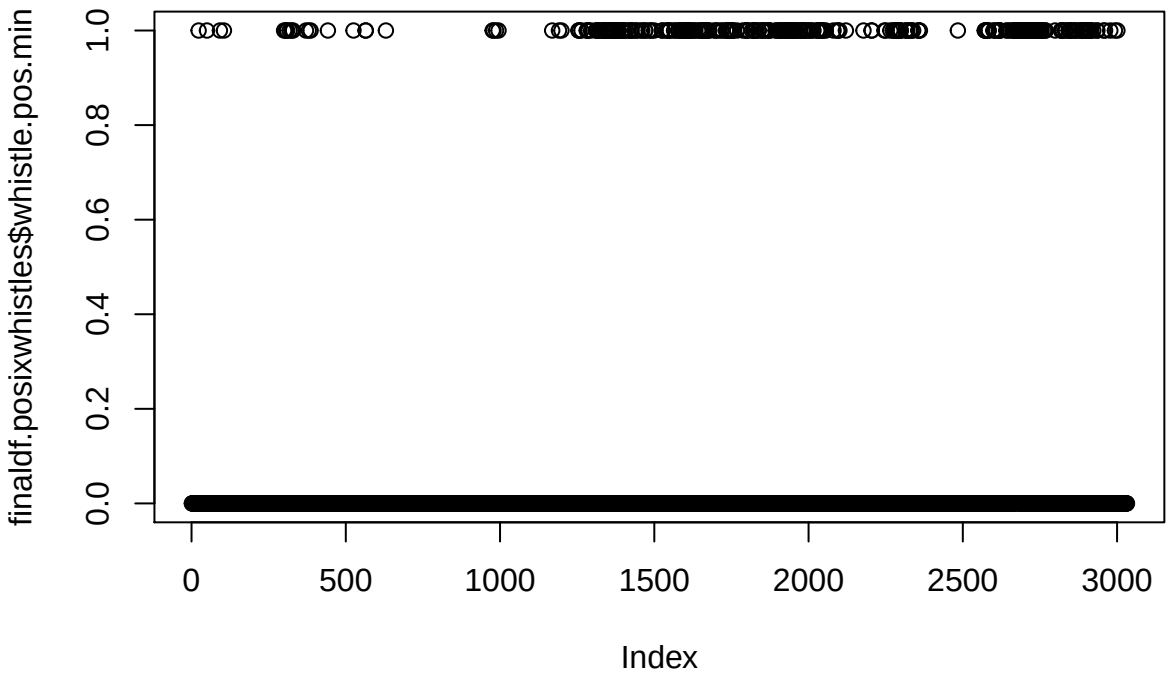
```
dat<- read.csv(file="C:/Users/emily/OneDrive/Documents/Masters/MScProject/Raw_data/Seconds_Data.csv")

dat$datetime <- paste0(dat$ï..StartDate," ", dat$StartTime)
dat$StartPosix <- as.POSIXct(dat$datetime, format="%Y:%m:%d %H:%M:%S", tz="UTC")
library(lubridate)

##time series in seconds
dat$EndPosix = dat$StartPosix + dat$DurationSeconds
dat$FloorPosixE<- floor_date(dat$EndPosix, unit="minute")

dat$FloorPosixS<-floor_date(dat$StartPosix, unit="minute")
dat$FloorPosixS
dat$Date <- as.Date(dat$StartPosix)
dat$Time <- format(as.POSIXct(dat$StartPosix), format = "%H:%M:%S", tz="UTC")
```

Creating 1 minute metric for whistles (whistle positive seconds)

```
dat$date <- paste0(dat$Date," ", dat$time)
dat$date<- as.POSIXct(dat$Date, format="%Y:%m:%d" )
dat$time <- format(as.POSIXct(dat$StartPosix), format = "%H:%M:%S")
dat$time

time.seq <- NA

for(y in 1:(nrow(dat)-1)){
  seq.tmp <- seq(from=dat$FloorPosixS[y], to=dat$FloorPosixE[y],by="min")
  seq.tmp <- as.character(seq.tmp)
  time.seq <- c(time.seq, seq.tmp)
  print(y)
  print(length(time.seq))
}

time.seq <- time.seq[-1]
```

Assigning whistle presence (Binary format) to 1 minute metric using previously filtered PAMGuard timestamped data

```
finaldf.posixwhistles <- data.frame(time= time.seq, whistle.pos.min=NA)
finaldf.posixwhistles$time <- as.POSIXct(finaldf.posixwhistles$time, format="%Y-%m-%d %H:%M:%S", tz="UT
positive.minutes <- unique(data1$posix)

finaldf.posixwhistles$whistle.pos.min <- ifelse(finaldf.posixwhistles$time %in% positive.minutes, 1, 0)

plot(finaldf.posixwhistles$whistle.pos.min)
```

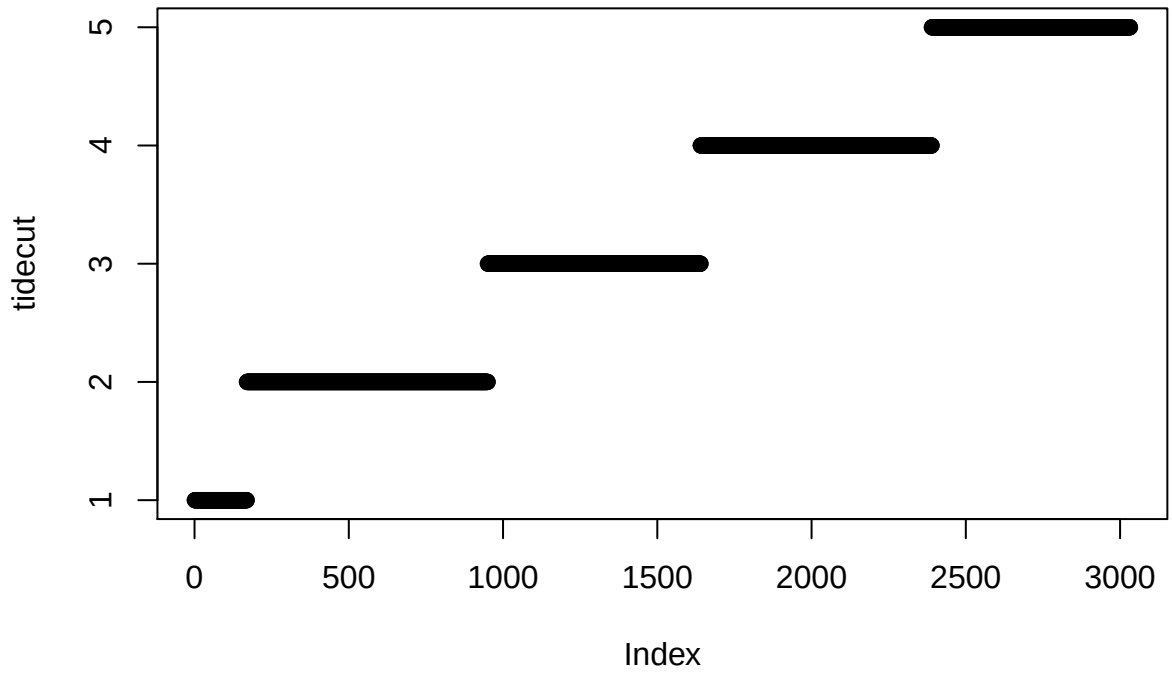



```
finaldf.posixwhistles$whistle.pos.min
finaldf.posixwhistles
```

Merging whistle positive minutes and high tide times

```
high_tide<- read.csv(file="C:/Users/emily/OneDrive/Documents/Tide.csv")
str(high_tide)

high_tide$datetime <- paste0(high_tide$ï..Date," ", high_tide$Time)
high_tide$Posix <- as.POSIXct(high_tide$datetime, format="%Y:%m:%d %H:%M:%S", tz="UTC")
```

```
high_tide$Posix[1]

tidecut<- cut(finaldf.posixwhistles$time, high_tide$Posix, labels = FALSE,
    include.lowest = FALSE, right = TRUE, dig.lab = 3,
    ordered_result = FALSE)
plot(tidecut)
```



Creating Time Since high Tide (TSHT) variable

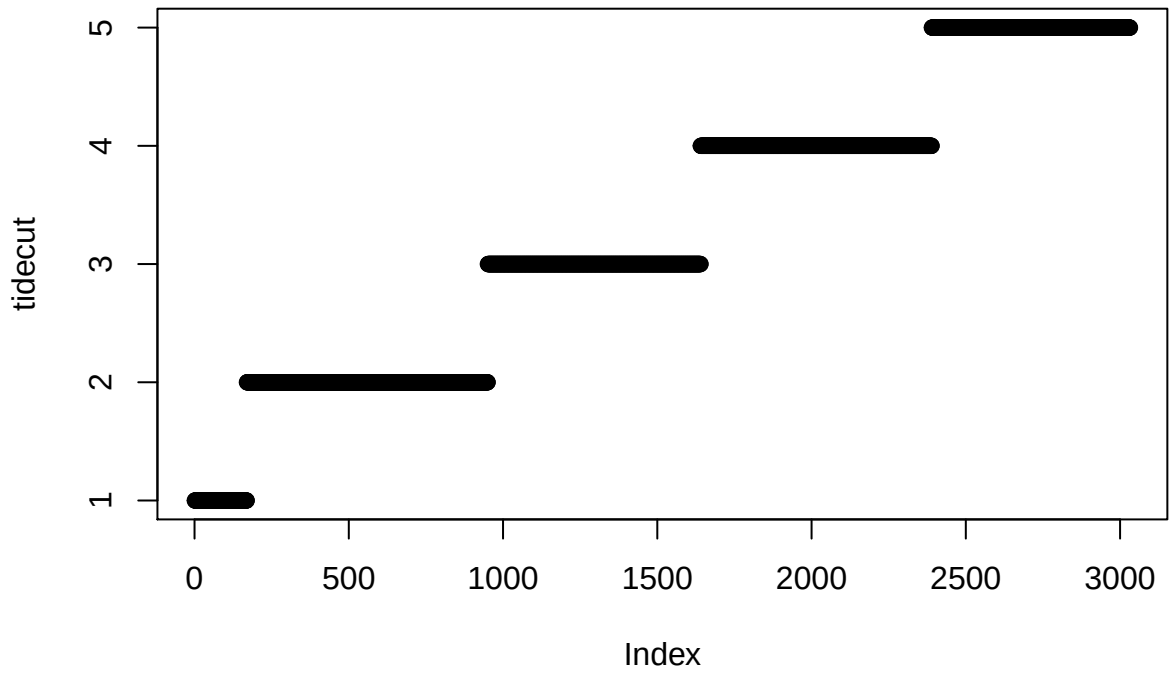
```
finaldf.posixwhistles$TSHT = finaldf.posixwhistles$time - high_tide$Posix[tidecut]
finaldf.posixwhistles$TSHT
```

Binning whistle presence by hour after high tide

```
hours= seq(0,13*60,60)
finaldf.posixwhistles$hrbins<- cut(as.numeric(finaldf.posixwhistles$TSHT), hours, labels = FALSE,
    include.lowest = FALSE, right = TRUE, dig.lab = 3,
    ordered_result = FALSE)
as.numeric(finaldf.posixwhistles$TSHT[50])
table(finaldf.posixwhistles$hrbins)
finaldf.posixwhistles$hrbins
```

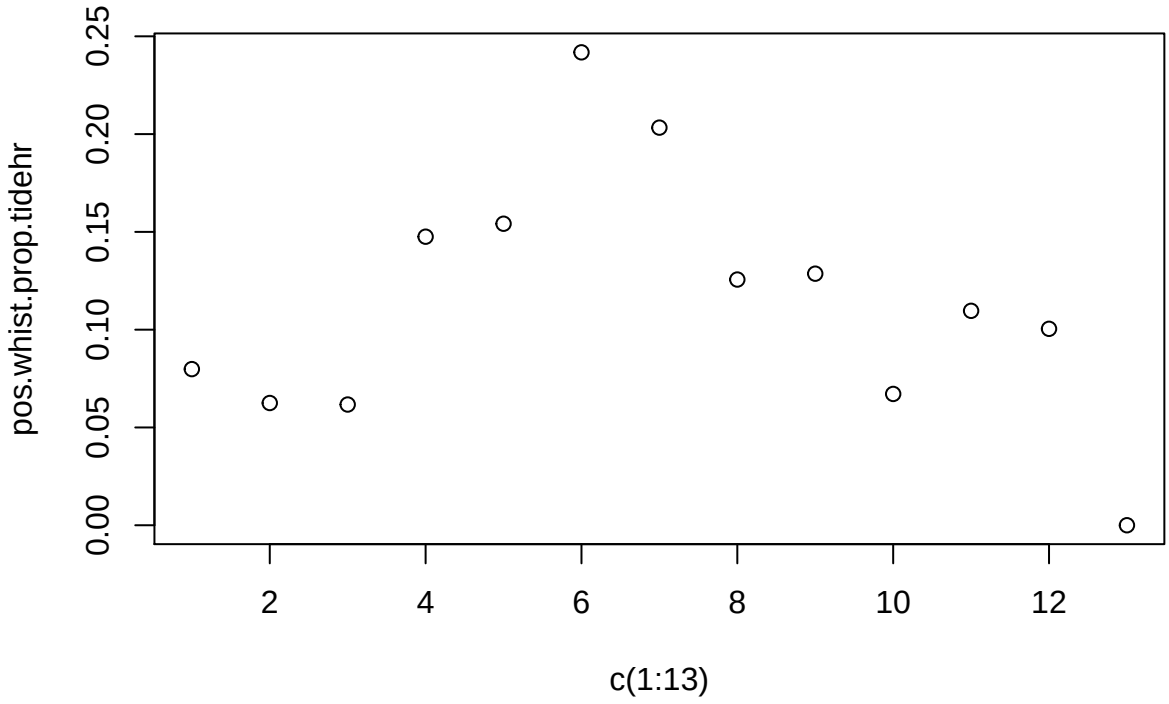
Separating tidal cycles for comparison in graph

```
finaldftide1=finaldf.posixwhistles[which(tidecut==1),]
finaldftide2=finaldf.posixwhistles[which(tidecut==2),]
finaldftide3=finaldf.posixwhistles[which(tidecut==3),]
finaldftide4=finaldf.posixwhistles[which(tidecut==4),]
finaldftide5=finaldf.posixwhistles[which(tidecut==5),]
plot(tidecut)
```



For all 5 tidal cycles across sample period

```
pos.whist.prop.tidehr=vector(,13)
for(y in 1:13){
  pos.whist.prop.tidehr[y]= sum(finaldf.posixwhistles$whistle.pos.min[which(finaldf.posixwhistles$hrbins
}
pos.whist.prop.tidehr
plot(x=c(1:13), y=pos.whist.prop.tidehr)
```



For tide cycle1

```
pos.whist.prop.tidehr1=vector(,13)
for(y in 1:13){
  pos.whist.prop.tidehr1[y]= sum(finaldftide1$whistle.pos.min[which(finaldftide1$hrbins==y)])/ length(f
}
pos.whist.prop.tidehr1
```

For tide cycle2

```
pos.whist.prop.tidehr2=vector(,13)
for(y in 1:13){
  pos.whist.prop.tidehr2[y]= sum(finaldftide2$whistle.pos.min[which(finaldftide2$hrbins==y)])/ length(f
}
pos.whist.prop.tidehr2
```

For tide cycle 3

```
pos.whist.prop.tidehr3=vector(,13)
for(y in 1:13){
```

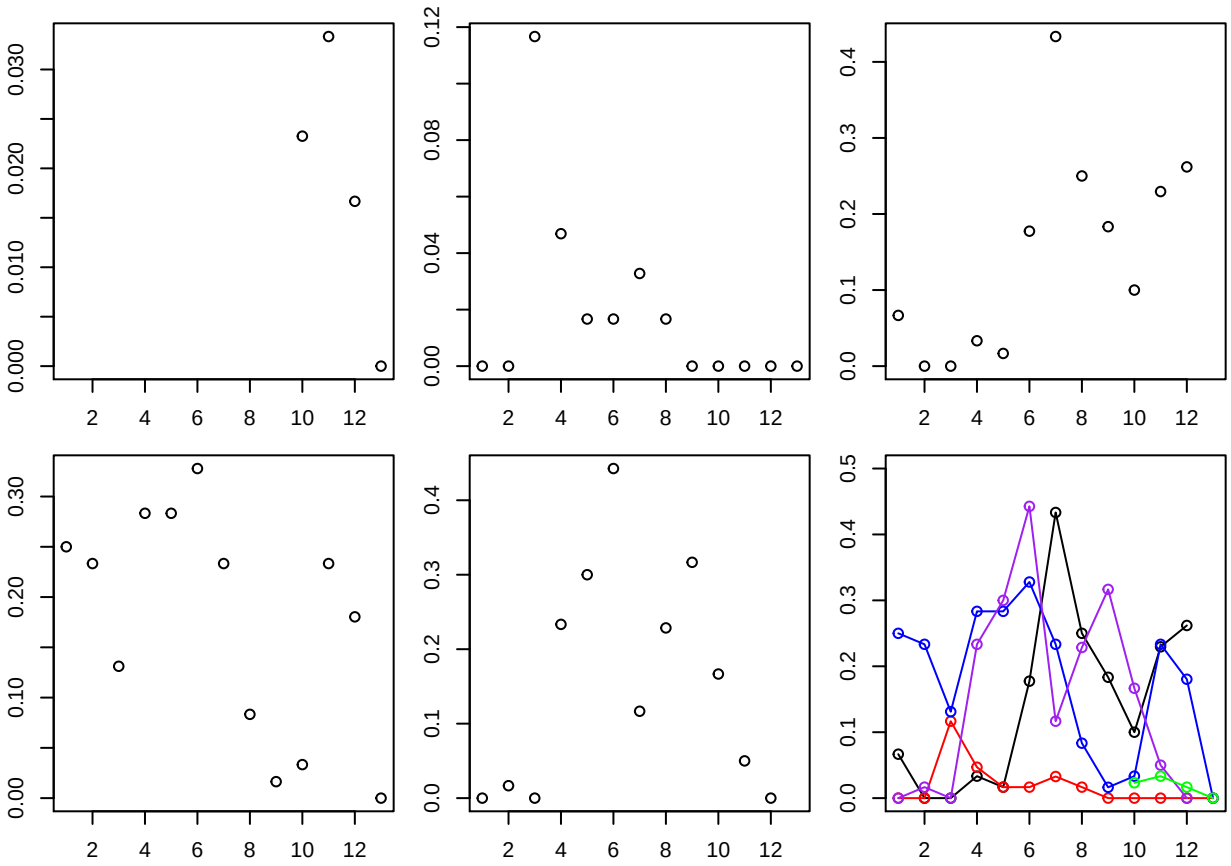
```
  pos.whist.prop.tidehr3[y]= sum(finaldftide3$whistle.pos.min[which(finaldftide3$hrbins==y)])/ length(fi
}
pos.whist.prop.tidehr3
```

For tide cycle 4

```
pos.whist.prop.tidehr4=vector(,13)
for(y in 1:13){
  pos.whist.prop.tidehr4[y]= sum(finaldftide4$whistle.pos.min[which(finaldftide4$hrbins==y)])/ length(fi
}
pos.whist.prop.tidehr4
```

For tide cycle 5

```
pos.whist.prop.tidehr5=vector(,13)
for(y in 1:13){
  pos.whist.prop.tidehr5[y]= sum(finaldftide5$whistle.pos.min[which(finaldftide5$hrbins==y)])/ length(fi
}
pos.whist.prop.tidehr5

par(mfrow= c(2,3))
par(mar= c(2,2,1,1))
plot(x=c(1:13), y=pos.whist.prop.tidehr1)
plot(x=c(1:13), y=pos.whist.prop.tidehr2)
plot(x=c(1:13), y=pos.whist.prop.tidehr3)
plot(x=c(1:13), y=pos.whist.prop.tidehr4)
plot(x=c(1:13), y=pos.whist.prop.tidehr5)

##final tidal plot (baseR)
plot(x=c(1:13), y=pos.whist.prop.tidehr3, lines(c(1:13),pos.whist.prop.tidehr3), ylim=  c(0,0.5), xlab=
points(pos.whist.prop.tidehr2, col="red",lines(c(1:13),pos.whist.prop.tidehr2,col="red"))
points(pos.whist.prop.tidehr4, col="blue",lines(c(1:13),pos.whist.prop.tidehr4, col="blue"))
points(pos.whist.prop.tidehr5, col="purple",lines(c(1:13),pos.whist.prop.tidehr5, col="purple"))
points(pos.whist.prop.tidehr1, col="green",lines(c(1:13),pos.whist.prop.tidehr1,col="green"))
```



Binning diel cycle for modeling

Changing time to decimal format for easier analysis

```
data1$Date <- as.Date(data1$posix)
finaldf.posixwhistles$posixtime <- format(as.POSIXct(finaldf.posixwhistles$time), format = "%H:%M:%S")
finaldf.posixwhistles$posixdate <- format(as.POSIXct(finaldf.posixwhistles$time), format = "%Y:%m:%d")

finaldf.posixwhistles$decDay = as.numeric(substr(finaldf.posixwhistles$posixdate,9,10)) +
   (as.numeric(substr(finaldf.posixwhistles$posixtime,1,2))/24) + (as.numeric(substr(finaldf.posixwhistl

finaldf.posixwhistles$decTime= as.numeric(substr(finaldf.posixwhistles$posixtime,1,2)) + (as.numeric(su
finaldf.posixwhistles$decTime
```

rounding down decimal time to hour

```
finaldf.posixwhistles$decTimebin<-floor(finaldf.posixwhistles$decTime)
finaldf.posixwhistles$decTimebin
```

rounding down decimal day so only shows date of recording

```
finaldf.posixwhistles$dec.dayfloor<- floor(finaldf.posixwhistles$decDay)
head(finaldf.posixwhistles)
```

separating days from finaldf.posixwhistles data frame

```
finaldfday1<- finaldf.posixwhistles[which(finaldf.posixwhistles$dec.dayfloor==7),]
finaldfday2<-finaldf.posixwhistles[which(finaldf.posixwhistles$dec.dayfloor==8),]
finaldfday3<-finaldf.posixwhistles[which(finaldf.posixwhistles$dec.dayfloor==9),]
```

Total diel cycle

```
pos.whist.prop.dielhr=vector(,23)
for(y in 1:23){
  pos.whist.prop.dielhr[y]= sum(finaldf.posixwhistles$whistle.pos.min[which(finaldf.posixwhistles$decTi
}
pos.whist.prop.dielhr
```

diel cycle of the 7th

```
pos.whist.prop.dielhr1=vector(,24)
for(y in 1:24){
  pos.whist.prop.dielhr1[y]= sum(finaldfday1$whistle.pos.min[which(finaldfday1$decTimebin==y)])/ length
}
head(finaldfday1)
```

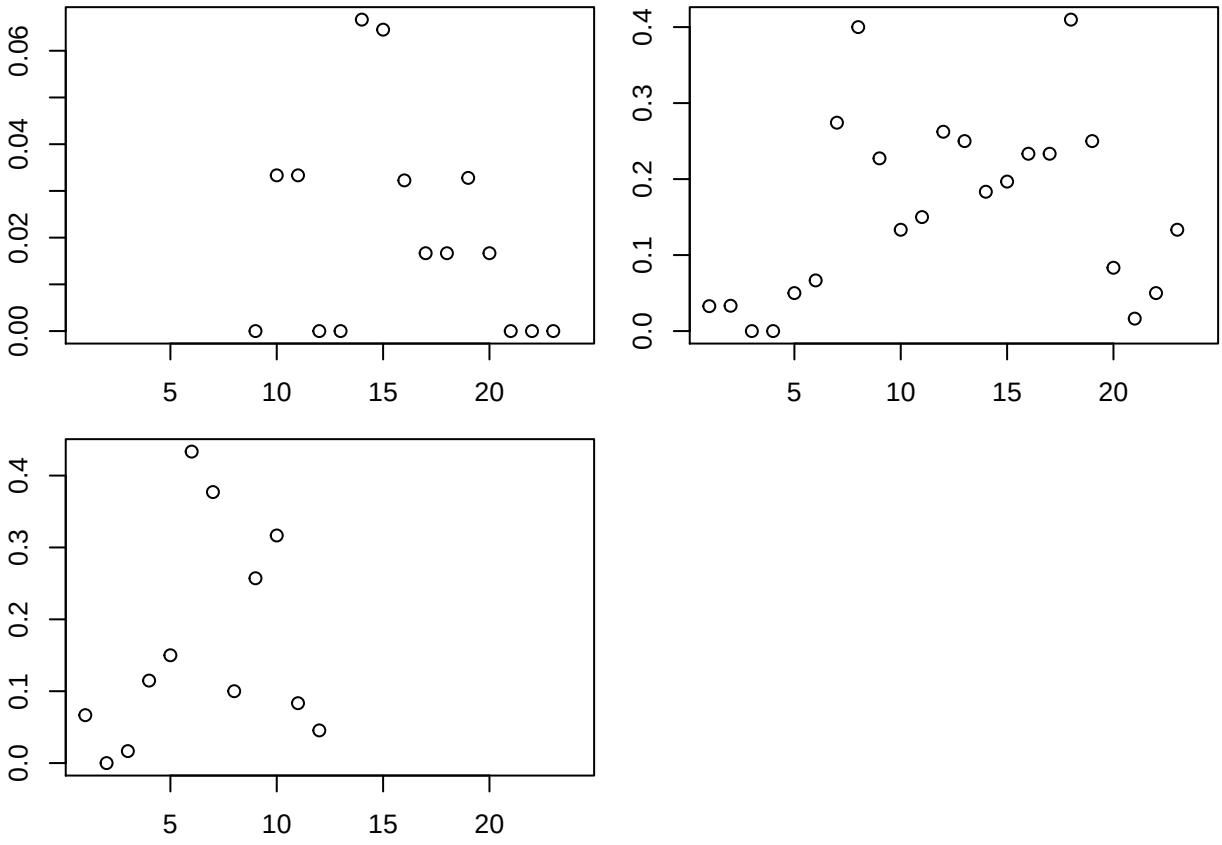
diel cycle of the 8th

```
pos.whist.prop.dielhr2=vector(,24)
for(y in 1:24){
  pos.whist.prop.dielhr2[y]= sum(finaldfday2$whistle.pos.min[which(finaldfday2$decTimebin==y)])/ length
}
```

diel cycle of the 9th

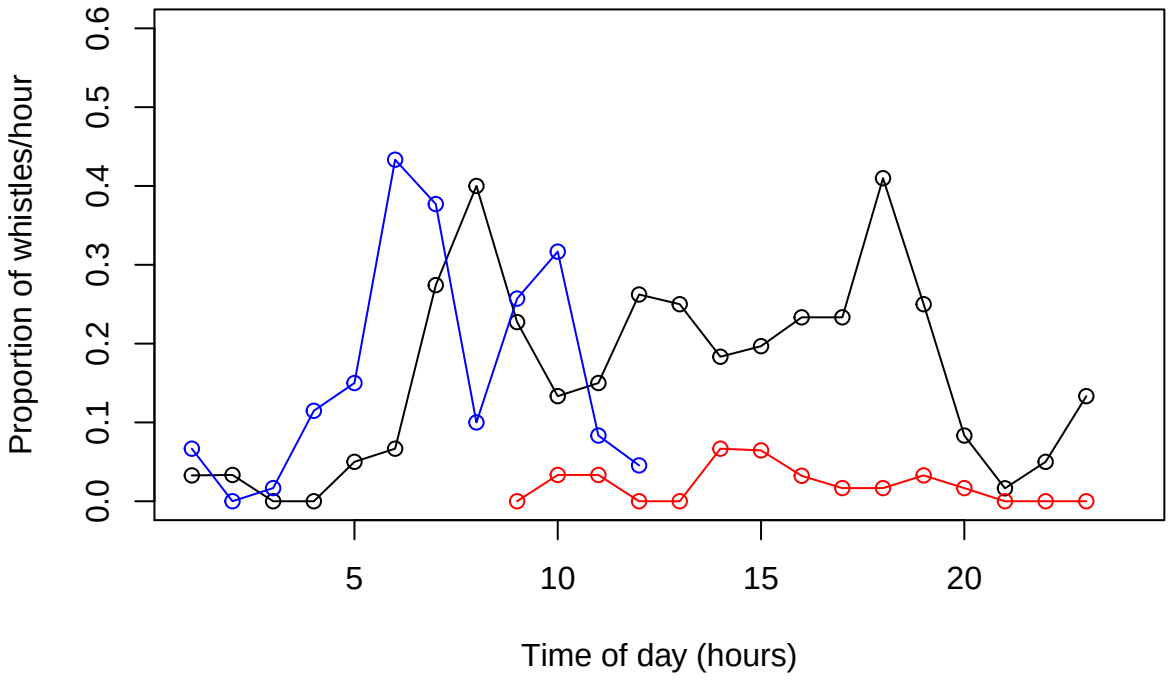
```
pos.whist.prop.dielhr3=vector(,24)
for(y in 1:24){
  pos.whist.prop.dielhr3[y]= sum(finaldfday3$whistle.pos.min[which(finaldfday3$decTimebin==y)])/ length
}
pos.whist.prop.dielhr2
par(mfrow= c(2,2))
```

```
par(mar= c(2,2,1,1))
plot(x=c(1:24), y=pos.whist.prop.dielhr1)
plot(x=c(1:24), y=pos.whist.prop.dielhr2)
plot(x=c(1:24), y=pos.whist.prop.dielhr3)
pos.whist.prop.dielhr2
```



final diel plot (baseR)

```
plot(x=c(1:24), y=pos.whist.prop.dielhr2, lines(c(1:24),pos.whist.prop.dielhr2), ylim=  c(0,0.6), xlab=
points(pos.whist.prop.dielhr1, col="red",lines(c(1:24),pos.whist.prop.dielhr1,col="red"))
points(pos.whist.prop.dielhr3, col="blue",lines(c(1:24),pos.whist.prop.dielhr3, col="blue"))
```



Adding and formatting boat data

```
library("schoolmath")
boats<- read.csv(file="C:/Users/emily/OneDrive/Documents/Masters/MScProject/Raw_data/Boat_data.csv")
##change to posix format
boats$StartPosix <- as.POSIXct(boats$Start.Time, format="%Y:%m:%d %H:%M:%S", tz="UTC")
boats$StopPosix<- as.POSIXct(boats$Stop.Time, format="%Y:%m:%d %H:%M:%S", tz="UTC")

##combining the stop_stop variable into 1 column
boatnoise_startstop<- c(as.POSIXct(boats$StartPosix, format= "%H:%M:%S", tz="UTC"), as.POSIXct(boats$St

##sorting start stop times in ascending order
sort.boat.noise<-sort(as.POSIXct(boatnoise_startstop, format="%Y:%m:%d %H:%M:%S", tz="UTC"))
sort.boat.noise
##binning total in relation to start stop time
finaldf.posixwhistles$boatcut<- cut(finaldf.posixwhistles$time, sort.boat.noise, labels = FALSE,
    include.lowest = FALSE, right = TRUE, dig.lab = 3,
    ordered_result = FALSE)
boatcut<- cut(finaldf.posixwhistles$time, sort.boat.noise, labels = FALSE,
    include.lowest = FALSE, right = TRUE, dig.lab = 3,
    ordered_result = FALSE)
##making binary format for boat presence/absence
boatpositive.seconds <- unique(sort.boat.noise)
finaldf.posixwhistles$boatnoise <- ifelse(finaldf.posixwhistles$time %in% boatpositive.seconds, 1, 0)
```

```
##assigning binary to boatboise
finaldf.posixwhistles$boatnoise= rep(0)
boatcut1<-if(is.na(boatcut)) {x=FALSE} else {if(boatcut) {boatcut}}
```

```
## Warning in if (is.na(boatcut)) {: the condition has length > 1 and only the
## first element will be used
```

```
## Warning in if (boatcut) {: the condition has length > 1 and only the first
## element will be used
```

```
boatcut=TRUE

finaldf.posixwhistles$boatnoise[which(is.odd(finaldf.posixwhistles$boatcut))]=1
```

Overall whistle proportion to boat presence

```
pos.whist.prop.boathr=vector(,2)
for(y in 0:1){
  pos.whist.prop.boathr[y+1]= sum(finaldf.posixwhistles$whistle.pos.min[which(finaldf.posixwhistles$boa
}
```

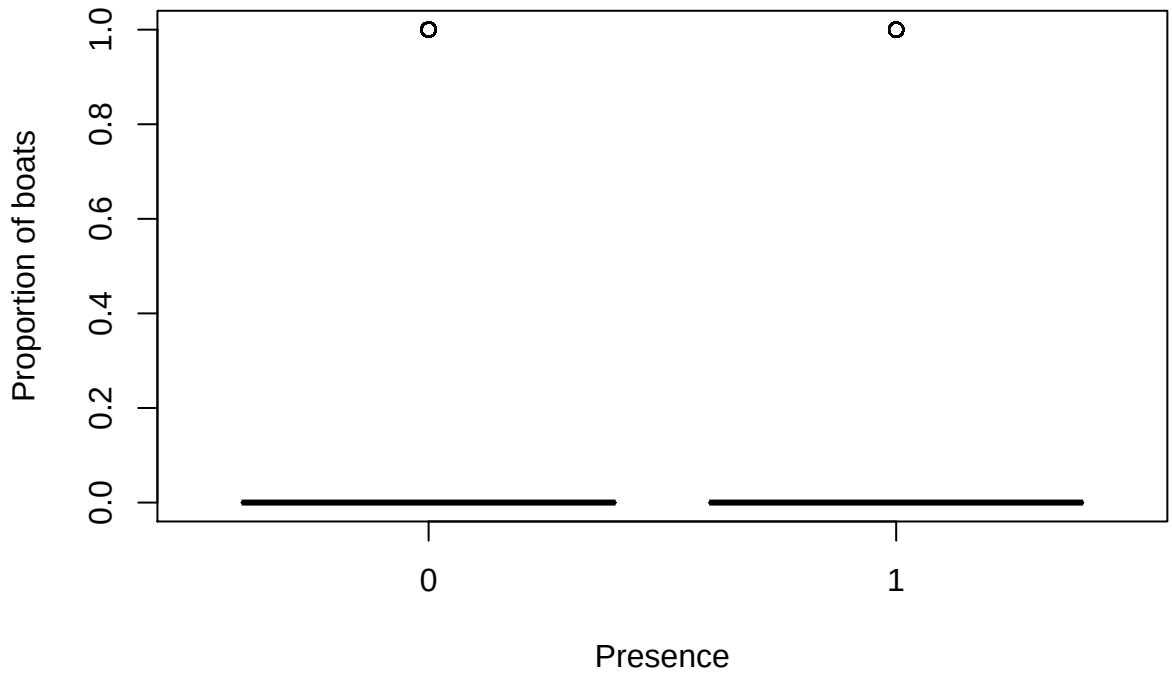
seperating boat noises to different periods in relation to whistle proportion

```
pos.whist.prop.boatsep=matrix(,max(boatcut),2)
for(y in 1:max(boatcut)){
  temp=finaldf.posixwhistles[which(finaldf.posixwhistles$boatcut==y),]
  pos.whist.prop.boatsep[y,1]= sum(temp$whistle.pos.min)/ length(temp$whistle.pos.min)
 pos.whist.prop.boatsep[y,2]=ifelse(is.odd(y),1,0)
}
```

Calculating boat presence in relation to whistles

```
boat.time.seq <- NA

for(y in 1:(nrow(boats)-1)){
  seq.tmp <- seq(from=boats$StartPosix[y], to=boats$StopPosix[y],by="min")
  seq.tmp <- as.character(seq.tmp)
  boat.time.seq <- c(boat.time.seq, seq.tmp)
  print(y)
  print(length(boat.time.seq))
}

boat.time.seq
boat.time.seq.uniq = unique(boat.time.seq)
boat.time.seq.uniq
```

```
boatdata<- as.data.frame(boat.time.seq.uniq)
colnames(boatdata)="time"
boatdata$time=as.POSIXct(boatdata$time, format="%Y-%m-%d %H:%M:%S", tz="UTC")
boatdata$boatpresence <- 1
boatdata

test=dplyr::left_join(finaldf.posixwhistles,boatdata,by="time")
test$boatpresence[is.na(test$boatpresence)]=0
test
test$Fboatpresence<- as.factor(test$boatpresence)
boxplot(whistle.pos.min~boatpresence, xlab="Presence", ylab= "Proportion of boats", data= test)
```

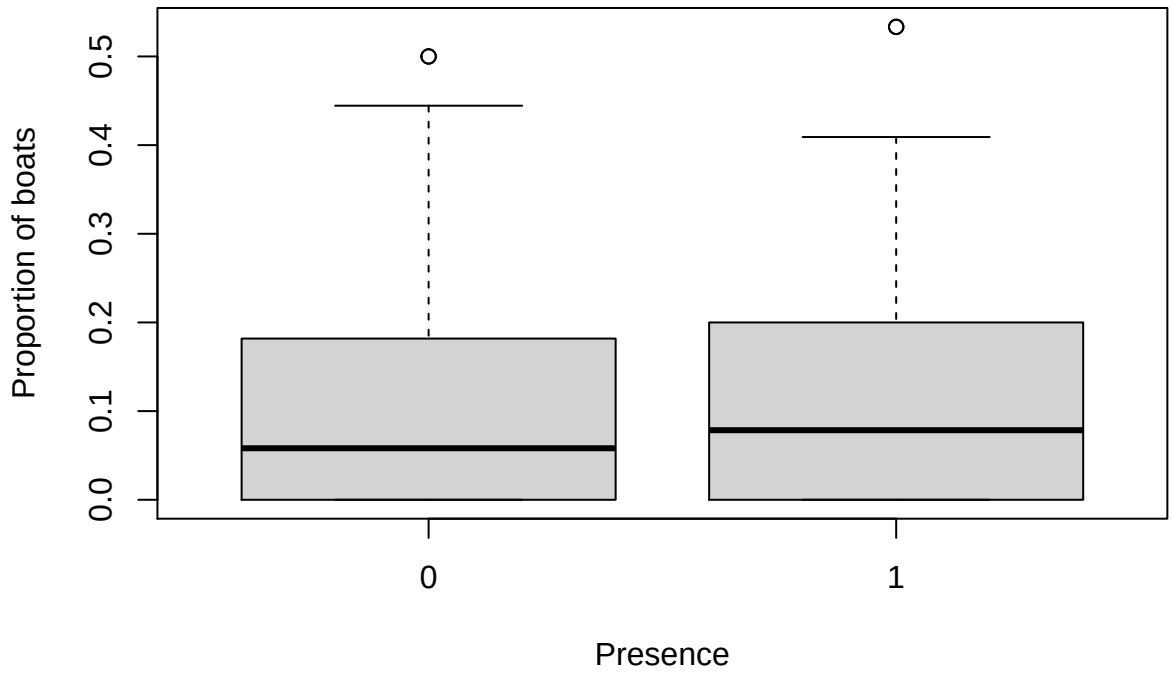



```
finaldf.posixwhistles$boatpresence<-test$boatpresence
finaldf.posixwhistles$Fboatpresence<-as.factor(test$boatpresence)
finaldf.posixwhistles
```

separate boat time distribution for boxplot

```
pos.whist.prop.boatsep=matrix(,max(finaldf.posixwhistles$boatcut),2)
for(y in 1:max(finaldf.posixwhistles$boatcut)){
  temp=finaldf.posixwhistles[which(finaldf.posixwhistles$boatcut==y),]
  pos.whist.prop.boatsep[y,1]= sum(temp$whistle.pos.min)/ length(temp$whistle.pos.min)
 pos.whist.prop.boatsep[y,2]=ifelse(is.odd(y),1,0)
```

```
}
pos.whist.prop.boatsep
boxplot(pos.whist.prop.boatsep[,1]~pos.whist.prop.boatsep[,2], xlab="Presence", ylab= "Proportion of boa
```



overall whistle proportion to boat presence

```
pos.whist.prop.boathr=vector(,2)
for(y in 0:1){
  pos.whist.prop.boathr[y+1]= sum(finaldf.posixwhistles$whistle.pos.min[which(finaldf.posixwhistles$boa
}
```

Putting diel cycle into ggplot form for report

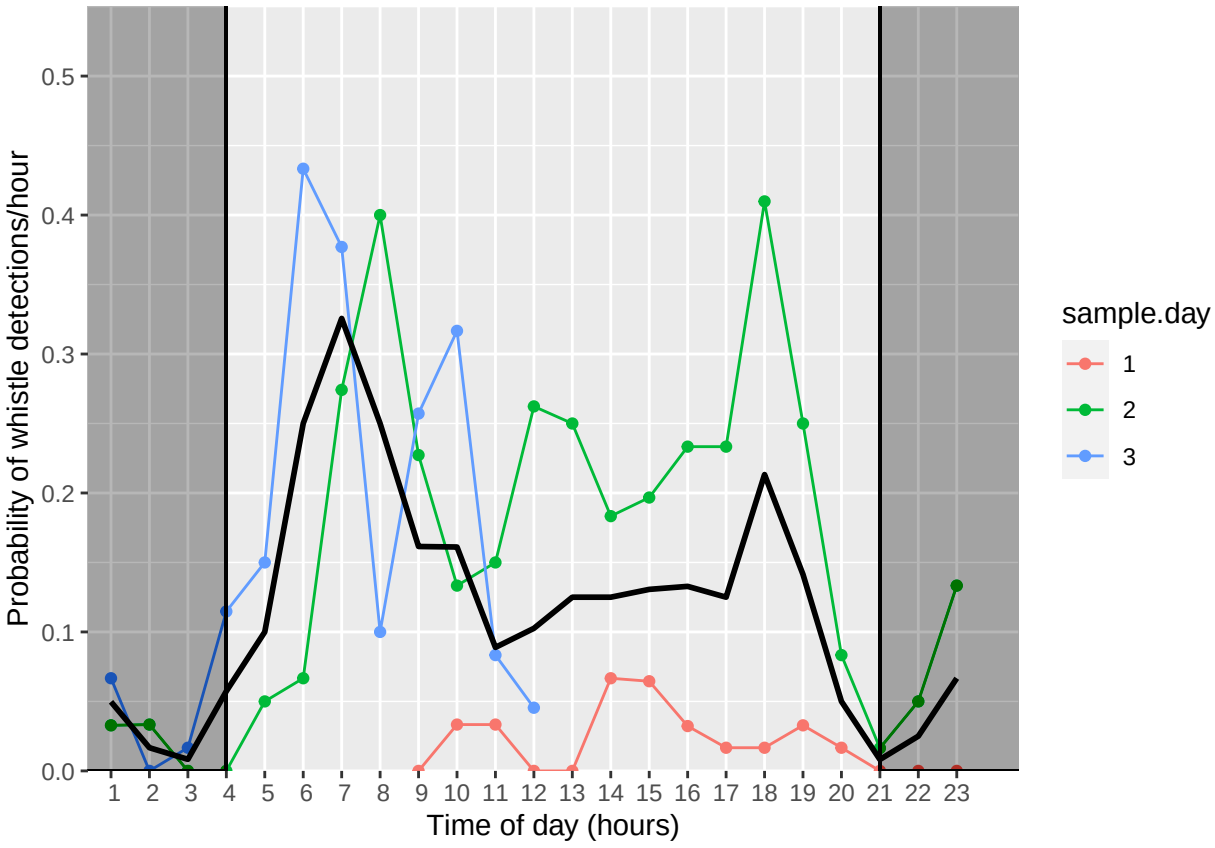
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
#creating dataframe for diel ggplot
time.of.day<- seq(1,24, 1)
sample.day<- rep(c(1,2,3),each=24)
```

```r
whistle.day<- c(pos.whist.prop.dielhr1,pos.whist.prop.dielhr2,pos.whist.prop.dielhr3)
dielgraphdf <- data.frame(whistle.day, time.of.day,sample.day)
dielgraphdf$sample.day<- as.factor(dielgraphdf$sample.day)
View(dielgraphdf)

ggplotday<-ggplot(dielgraphdf, aes(x = time.of.day, y = whistle.day, colour = sample.day, group = sampl
  geom_point()+
  geom_line()
ggplotday+labs(x ="Time of day (hours)", y = "Probability of whistle detections/hour") + scale_x_discre
    fill="black", colour= "black", alpha=0.005,
    xmin=0,
    xmax=4,
    ymin=0,
    ymax=2
    ) + scale_x_discrete(limits=c(1:23)) +  geom_rect(
    fill="black", colour= "black", alpha=0.005,
    xmin=21,
    xmax=25,
    ymin=0,
    ymax=2
    )+
  scale_y_continuous(limits = c(0,0.55), expand = c(0, 0))+stat_summary(fun.y=mean,geom="line", colour=
```

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?

## Warning: Continuous limits supplied to discrete scale.
## Did you mean `limits = factor(...)` or `scale_*_continuous()`?

## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: `fun.y` is deprecated. Use `fun` instead.

## Warning: Removed 22 rows containing non-finite values (stat_summary).

## Warning: Removed 22 rows containing missing values (geom_point).

## Warning: Removed 22 row(s) containing missing values (geom_path).
```



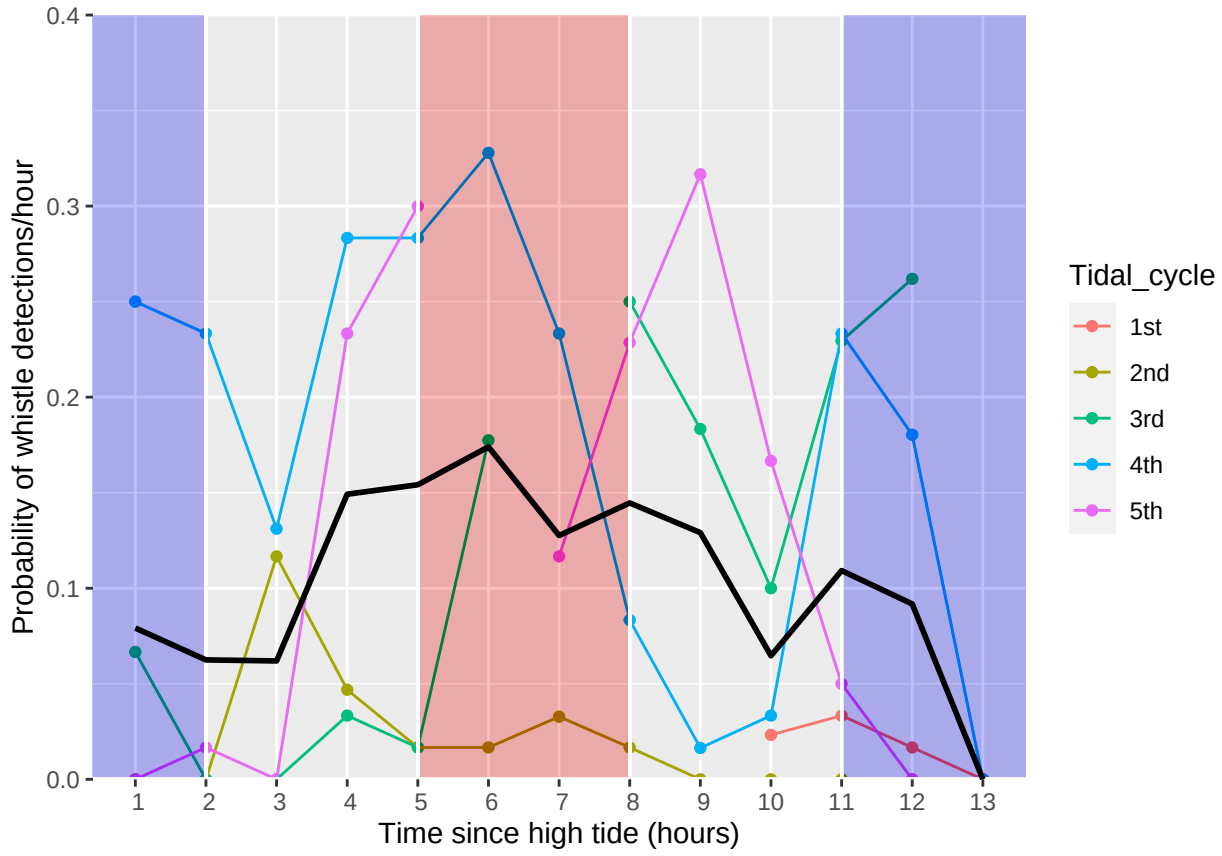
Putting time since high tide into ggplot form for report

```
#creating dataframe for tide ggplot
time.since.ht<- seq(1,13, 1)
tidecycle<- rep(c("1st","2nd","3rd","4th","5th"),each=13)

whistle.tide<-c(pos.whist.prop.tidehr1,pos.whist.prop.tidehr2,pos.whist.prop.tidehr3,pos.whist.prop.tid
tidegraphdf<- data.frame(whistle.tide, time.since.ht,tidecycle)
tidegraphdf$Tidal_cycle<- as.factor(tidegraphdf$tidecycle)
View(tidegraphdf)

ggplottide<-ggplot(tidegraphdf, aes(x = time.since.ht, y = whistle.tide, colour = Tidal_cycle, group =
  geom_point(size=1.5)+
  geom_line(alpha=5)
ggplottide + labs(x="Time since high tide (hours)", y="Probability of whistle detections/hour") + scale
    fill="blue",colour= "white",alpha=0.002,
    xmin=0,
    xmax=2,
    ymin=0,
    ymax=2
    ) + scale_x_discrete(limits=c(1:13)) +  geom_rect(
    fill="white",colour= "white",alpha=0.0025,
    xmin=2,
    xmax=5,
```

```
    ymin=0,
    ymax=2
    ) + scale_x_discrete(limits=c(1:13)) +  geom_rect(
    fill="red",colour= "white",alpha=0.002,
    xmin=5,
    xmax=8,
    ymin=0,
    ymax=2
    ) + scale_x_discrete(limits=c(1:13)) +  geom_rect(
    fill="white",colour= "white",alpha=0.0025,
    xmin=8,
    xmax=11,
    ymin=0,
    ymax=2
    ) + scale_x_discrete(limits=c(1:13)) +  geom_rect(
    fill="blue",colour= "white",alpha=0.002,
    xmin=11,
    xmax=14,
    ymin=0,
    ymax=2
    ) +
  scale_y_continuous(limits = c(0,0.4), expand = c(0, 0))+stat_summary(fun.y=mean,geom="line", colour="
```

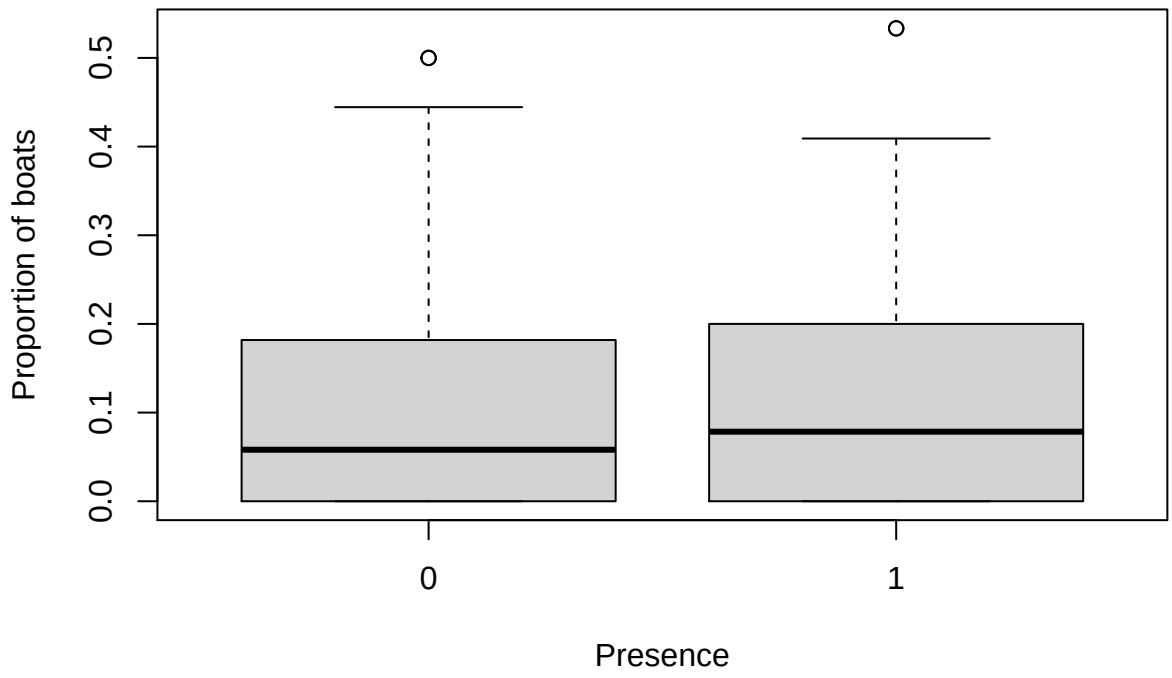
```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?

## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?

## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?

## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?

## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?

## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Warning: 'fun.y' is deprecated. Use 'fun' instead.
```

```
## Warning: Removed 13 rows containing non-finite values (stat_summary).

## Warning: Removed 13 rows containing missing values (geom_point).

## Warning: Removed 11 row(s) containing missing values (geom_path).
```



Putting boat presence into ggplot form for report

```
library(RColorBrewer)
library(ggplot2)
boxplot(pos.whist.prop.boatsep[,1]~pos.whist.prop.boatsep[,2], xlab="Presence", ylab= "Proportion of boa
```




```
colnames(pos.whist.prop.boatsep) <- c("Boat.prop", "Boat Presence")
presence.cat<- rep(c(1,0),times=61)
boatboxdf<- data.frame(pos.whist.prop.boatsep)
pos.whist.prop.boatsep
```

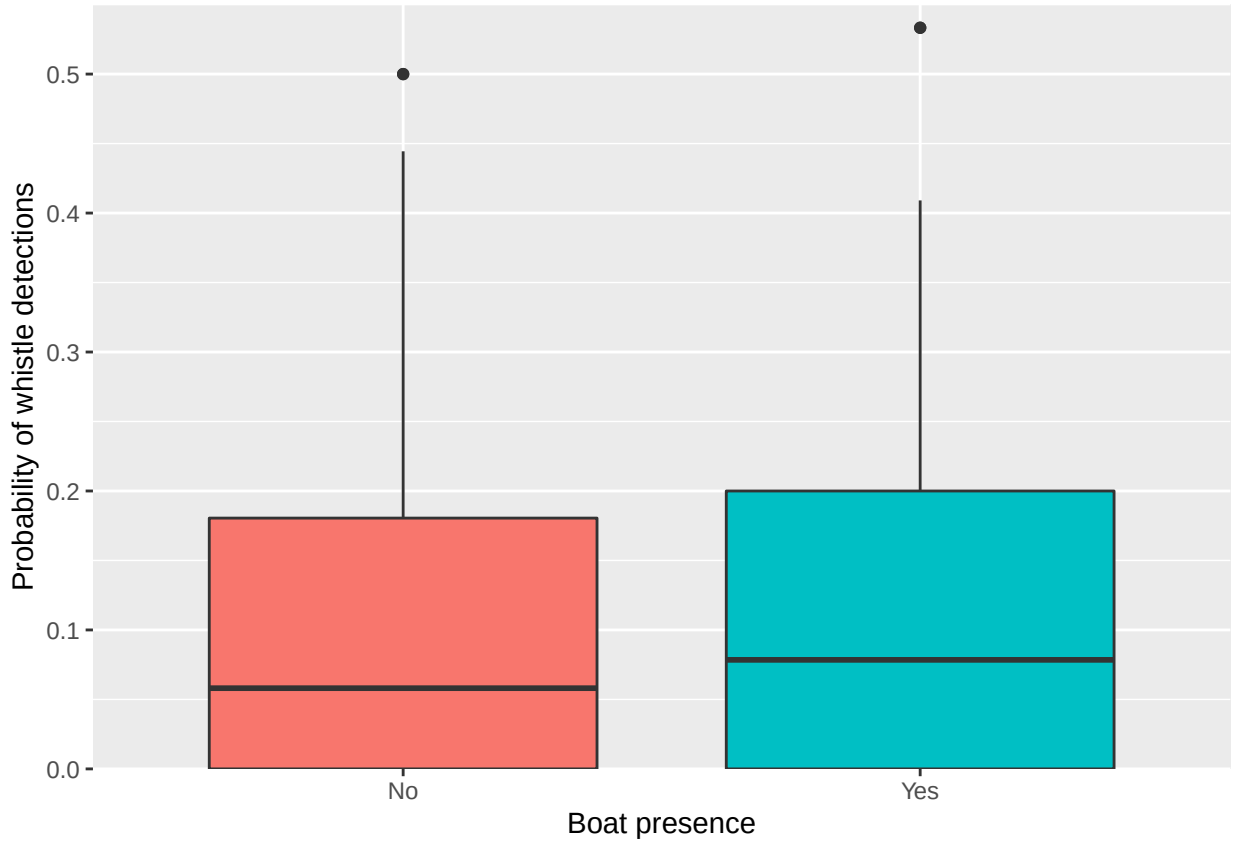
```
##          Boat.prop Boat Presence
##    [1,]        NaN             1
##    [2,]        NaN             0
##    [3,]        NaN             1
##    [4,] 0.02857143             0
##    [5,] 0.00000000             1
##    [6,] 0.08695652             0
##    [7,] 0.00000000             1
##    [8,] 0.00000000             0
##    [9,] 0.00000000             1
##   [10,] 0.00000000             0
##   [11,] 0.00000000             1
##   [12,] 0.00000000             0
##   [13,] 0.00000000             1
##   [14,] 0.03125000             0
##   [15,] 0.20000000             1
##   [16,] 0.08695652             0
##   [17,] 0.05128205             1
##   [18,] 0.02857143             0
##   [19,] 0.03448276             1
```

```
##  [20,] 0.00000000            0
##  [21,] 0.00000000            1
##  [22,] 0.01111111            0
##  [23,] 0.08000000            1
##  [24,] 0.00000000            0
##  [25,] 0.03125000            1
##  [26,] 0.00000000            0
##  [27,] 0.00000000            1
##  [28,] 0.00000000            0
##  [29,] 0.00000000            1
##  [30,] 0.00000000            0
##  [31,] 0.00000000            1
##  [32,] 0.00000000            0
##  [33,] 0.00000000            1
##  [34,] 0.00000000            0
##  [35,] 0.00000000            1
##  [36,] 0.00000000            0
##  [37,] 0.00000000            1
##  [38,] 0.06060606            0
##  [39,] 0.00000000            1
##  [40,] 0.00000000            0
##  [41,] 0.00000000            1
##  [42,] 0.01030928            0
##  [43,] 0.00000000            1
##  [44,] 0.10000000            0
##  [45,] 0.06666667            1
##  [46,] 0.00000000            0
##  [47,] 0.00000000            1
##  [48,] 0.00000000            0
##  [49,] 0.13636364            1
##  [50,] 0.16666667            0
##  [51,] 0.17391304            1
##  [52,] 0.17647059            0
##  [53,] 0.40000000            1
##  [54,] 0.50000000            0
##  [55,] 0.28571429            1
##  [56,] 0.24137931            0
##  [57,] 0.26666667            1
##  [58,] 0.16666667            0
##  [59,] 0.13333333            1
##  [60,] 0.15277778            0
##  [61,] 0.17142857            1
##  [62,] 0.24539877            0
##  [63,] 0.12500000            1
##  [64,] 0.20000000            0
##  [65,] 0.26666667            1
##  [66,] 0.14285714            0
##  [67,] 0.40000000            1
##  [68,] 0.20000000            0
##  [69,] 0.26666667            1
##  [70,] 0.21739130            0
##  [71,] 0.53333333            1
##  [72,] 0.33333333            0
##  [73,] 0.35294118            1
```

```
##  [74,] 0.22000000            0
##  [75,] 0.40000000            1
##  [76,] 0.18181818            0
##  [77,] 0.00000000            1
##  [78,] 0.00000000            0
##  [79,] 0.12903226            1
##  [80,] 0.00000000            0
##  [81,] 0.07692308            1
##  [82,] 0.00000000            0
##  [83,] 0.00000000            1
##  [84,] 0.03921569            0
##  [85,] 0.04000000            1
##  [86,] 0.05555556            0
##  [87,] 0.06666667            1
##  [88,] 0.27906977            0
##  [89,] 0.20000000            1
##  [90,] 0.00000000            0
##  [91,] 0.26666667            1
##  [92,] 0.00000000            0
##  [93,] 0.16000000            1
##  [94,] 0.00000000            0
##  [95,] 0.00000000            1
##  [96,] 0.06081081            0
##  [97,] 0.13333333            1
##  [98,] 0.22222222            0
##  [99,] 0.13333333            1
## [100,] 0.09090909            0
## [101,] 0.53333333            1
## [102,] 0.44444444            0
## [103,] 0.40909091            1
## [104,] 0.08108108            0
## [105,] 0.13043478            1
## [106,] 0.27272727            0
## [107,] 0.15384615            1
## [108,]        NaN            0
## [109,]        NaN            1
## [110,]        NaN            0
## [111,]        NaN            1
## [112,] 0.50000000            0
## [113,] 0.26666667            1
## [114,] 0.17177914            0
## [115,] 0.00000000            1
```

```
colnames(boatboxdf)<- c("Boat.prop", "Presence")
boatboxdf$Presence<- as.factor(boatboxdf$Presence)

ggplot(boatboxdf, aes(x=Presence, y=Boat.prop, fill=Presence))+
         geom_boxplot(alpha=1) +
    theme(legend.position="none") + labs(x="Boat presence", y="Probability of whistle detections")+   s
```

```
## Warning: Removed 7 rows containing non-finite values (stat_boxplot).
```

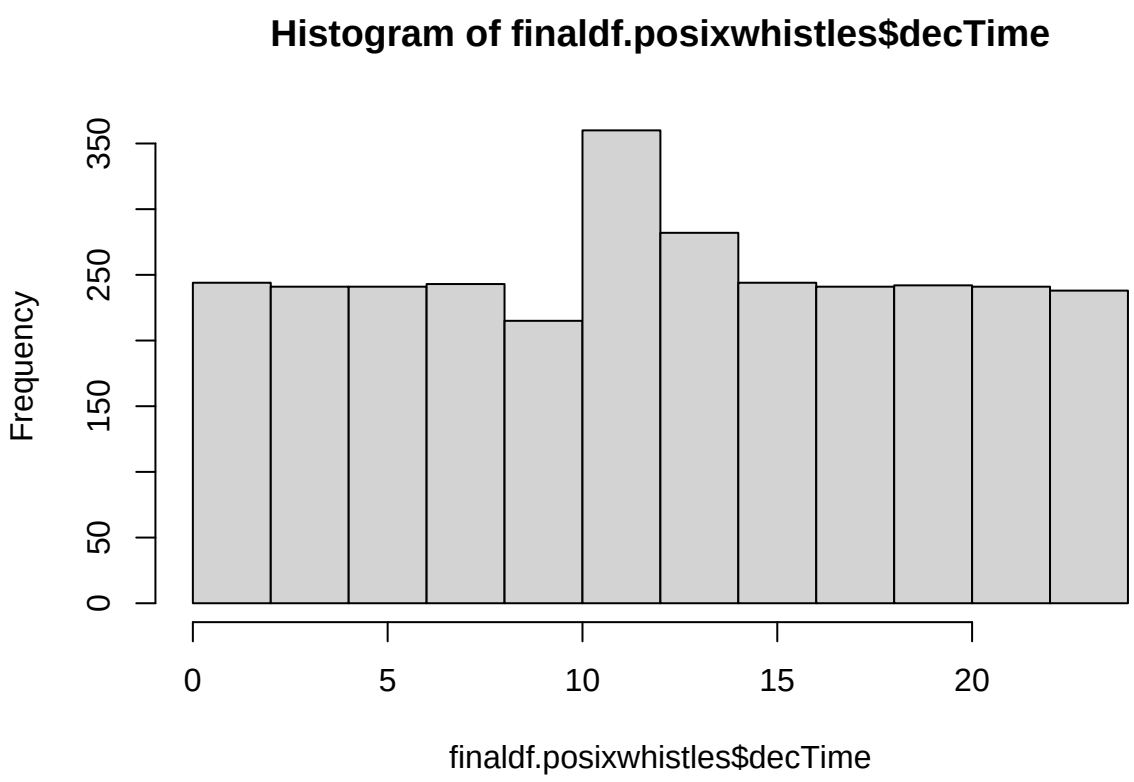


Whistle detection probability descriptive statistics

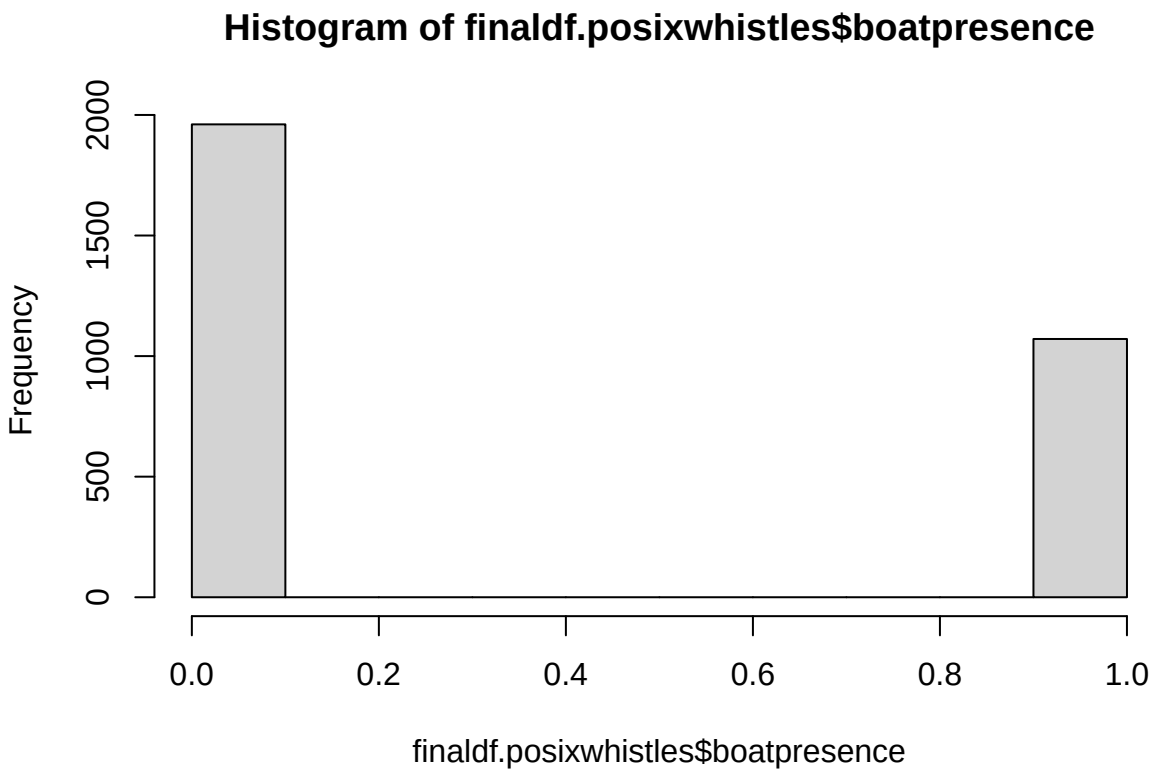
```
require(plotrix)
```

```
## Loading required package: plotrix
```

```
library(plotrix)
##diel cycle
std.error(pos.whist.prop.dielhr)
summary(pos.whist.prop.dielhr)
sd(pos.whist.prop.dielhr)
quantile(pos.whist.prop.dielhr)
hist(finaldf.posixwhistles$decTime)
```

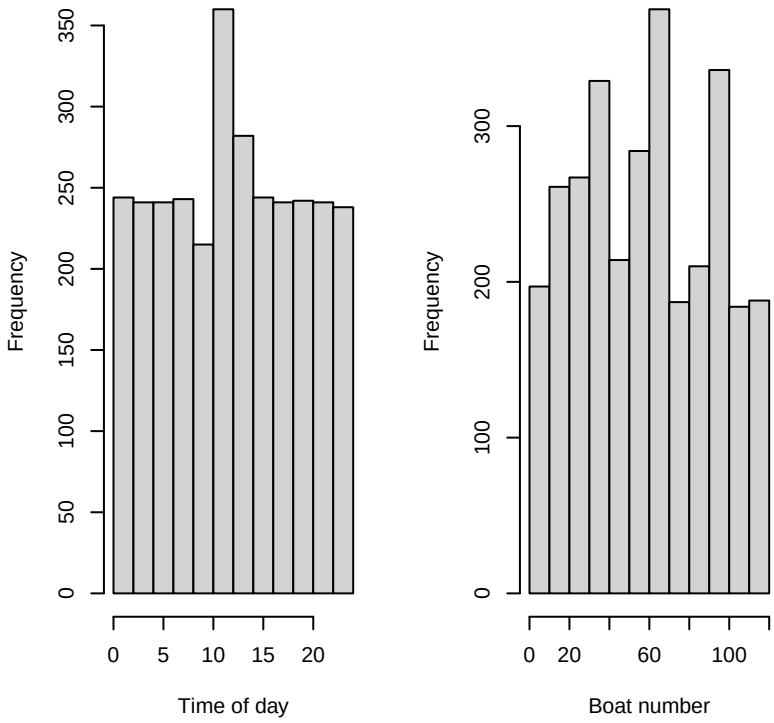



```
##tidal cycle
std.error(pos.whist.prop.tidehr)
summary(pos.whist.prop.tidehr)
sd(pos.whist.prop.tidehr)
quantile(pos.whist.prop.tidehr)

####hist(finaldf.posixwhistles$as.n.TSHT)<- wont knit properly for some reason

##boat presence
std.error(pos.whist.prop.boathr)
summary(pos.whist.prop.boathr)
sd(pos.whist.prop.boathr)
quantile(pos.whist.prop.boathr)
hist(finaldf.posixwhistles$boatpresence)
```



Distributions of environmental variable data

```
par(mfrow = c(1,3))
hist(finaldf.posixwhistles$decTime, xlab="Time of day", main=NULL)

####hist(finaldf.posixwhistles$as.n.TSHT, xlab="Time since high tide (minutes)",main=NULL)<- wont knit

hist(finaldf.posixwhistles$boatcut, xlab="Boat number",main=NULL)
```



GAM modeling

making sure model cannot be fitted with linear model

```
library(mgcv)
```

```
## Warning: package 'mgcv' was built under R version 4.0.5
```

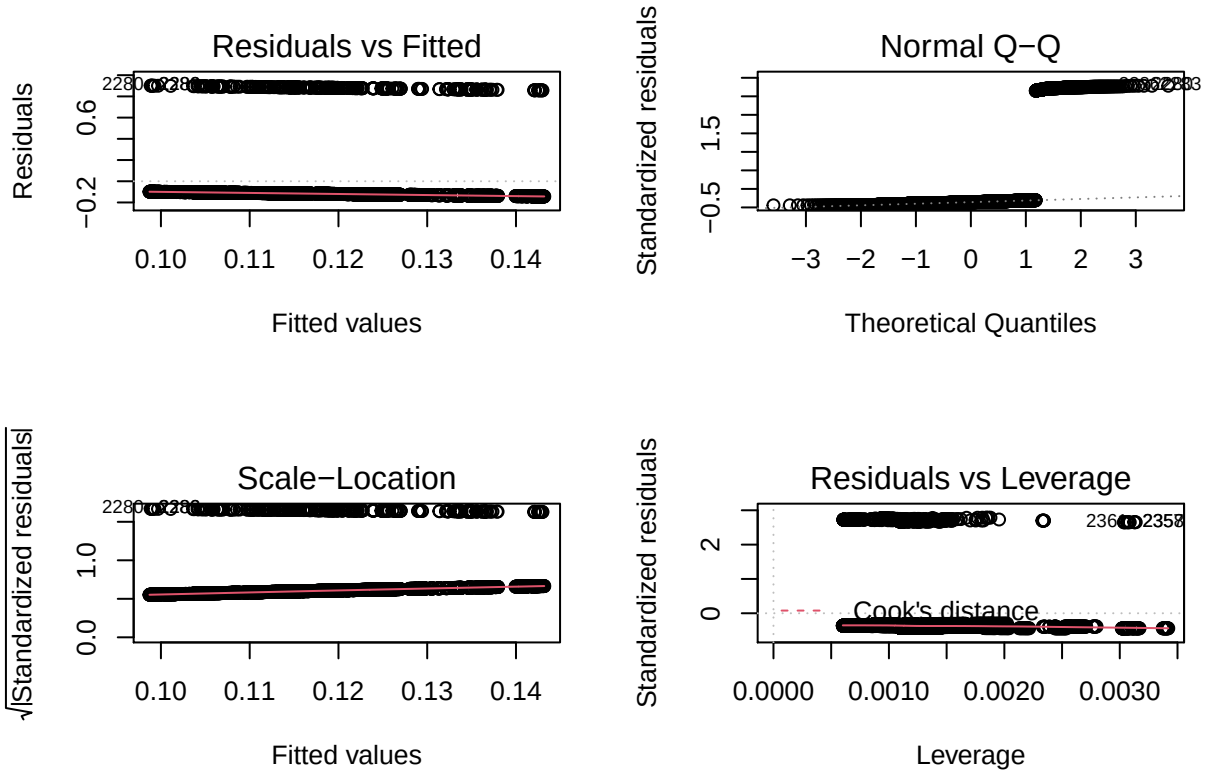
```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.8-36. For overview type 'help("mgcv-package")'.
```

```
lm1= lm(whistle.pos.min~ Fboatpresence+ TSHT +decTime, data=finaldf.posixwhistles)

par(mfrow = c(2,2))
plot(lm1)
```



Through model validation- data should be put into GAM model as relationship is non-linear, no normality line, residuals vs fitted are grouped together not equally spread, observations clearly dependent

First GAM model construction-with all environmental variables

```
finaldf.posixwhistles$as.n.TSHT<- as.numeric(finaldf.posixwhistles$TSHT)

gam1<- gam(whistle.pos.min~ Fboatpresence+ s(as.n.TSHT,bs="cc") + s(decTime ,bs="cc",k=24),family="bino

gam1
```

```
##diel variable removed
gam1.1<- gam(whistle.pos.min~ Fboatpresence+ s(as.n.TSHT,bs="cc"),family="binomial", data=finaldf.posix
##tide variable removed
gam1.2<- gam(whistle.pos.min~ Fboatpresence+ s(decTime ,bs="cc",k=24),family="binomial", data=finaldf.p
##boat variable removed
gam1.3<-gam(whistle.pos.min~ + s(as.n.TSHT,bs="cc") + s(decTime ,bs="cc",k=24),family="binomial", data=
```

plotting variable binned data in relation to GAM model

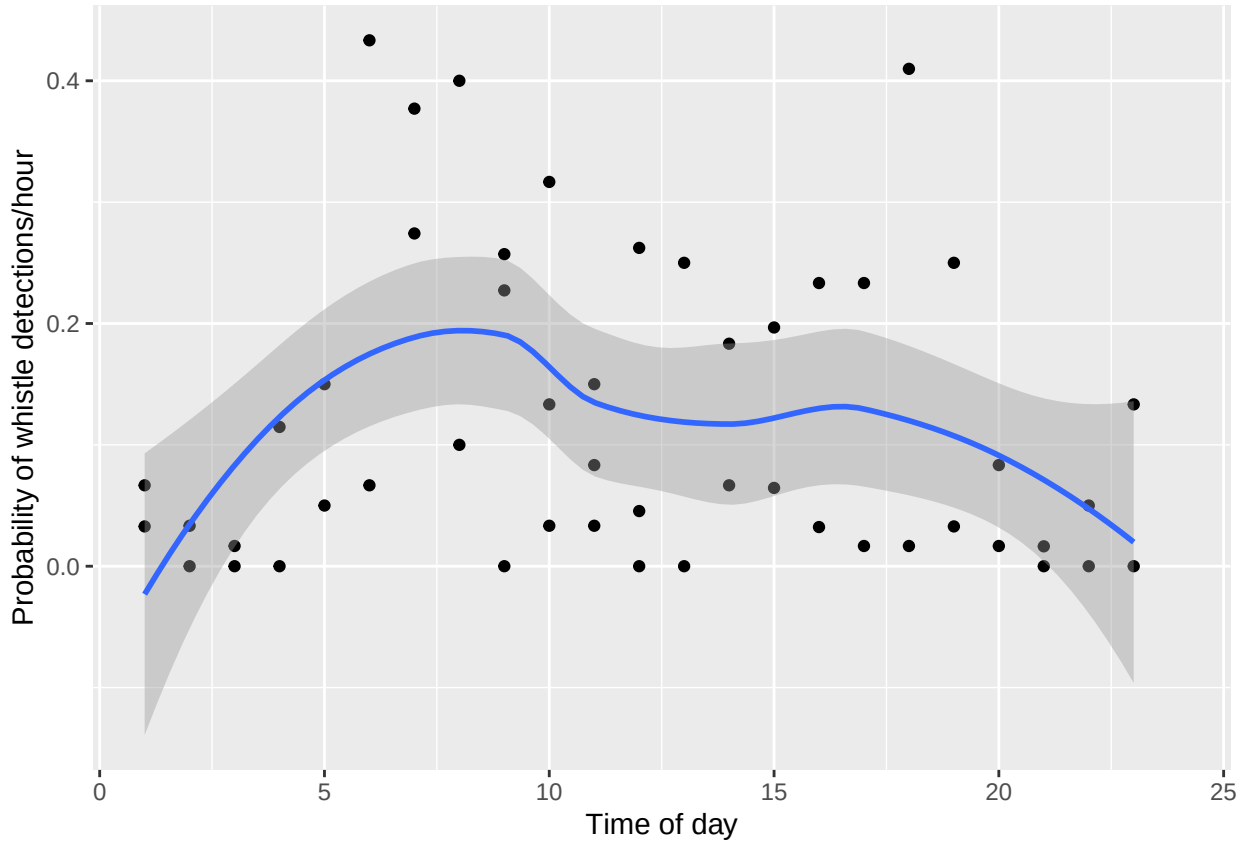
diel

```
gg.day.gam<-ggplot(dielgraphdf, aes(x = time.of.day, y = whistle.day))+ geom_point() + geom_smooth()
gg.day.gam +labs(x="Time of day", y="Probability of whistle detections/hour")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 22 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 22 rows containing missing values (geom_point).
```



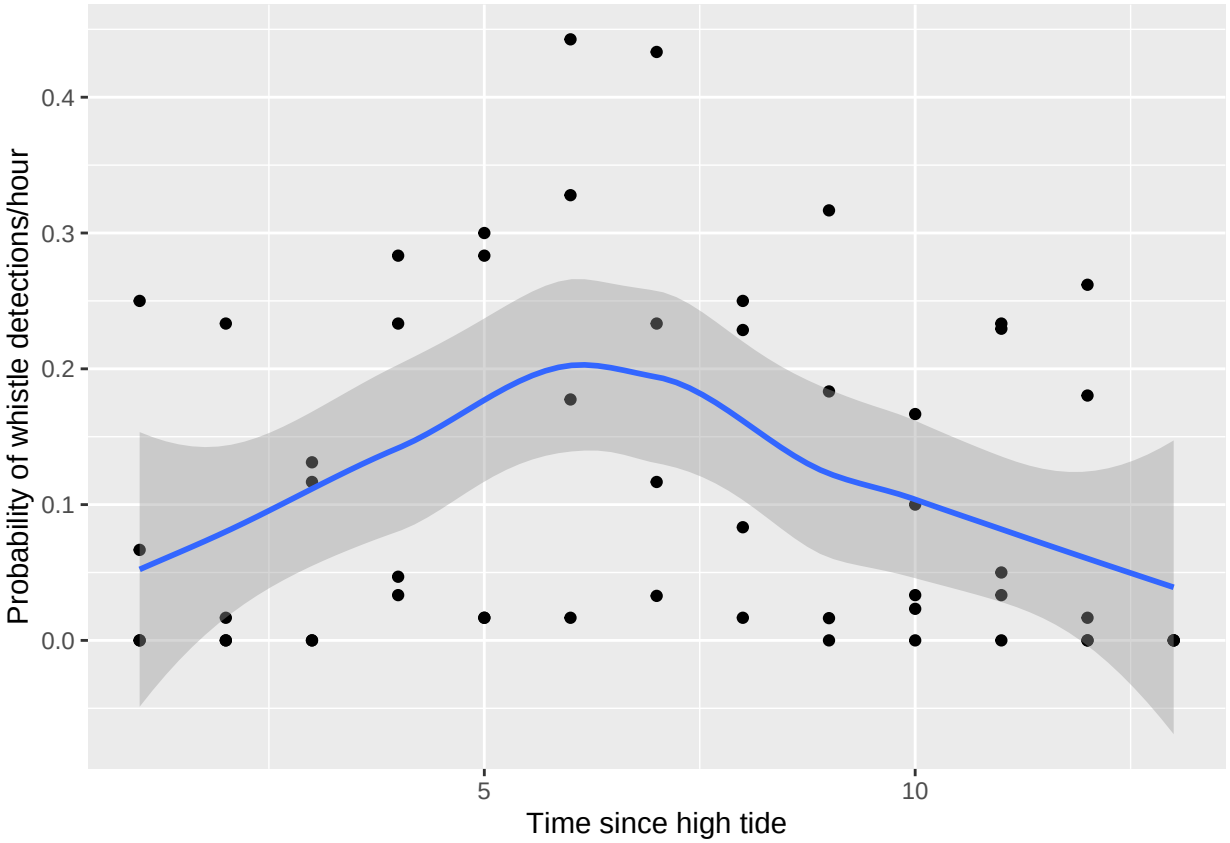
Tide

```
gg.tide.gam<-ggplot(tidegraphdf, aes(x = time.since.ht, y = whistle.tide))+ geom_point() + geom_smooth()
gg.tide.gam +labs(x="Time since high tide", y="Probability of whistle detections/hour")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 11 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```



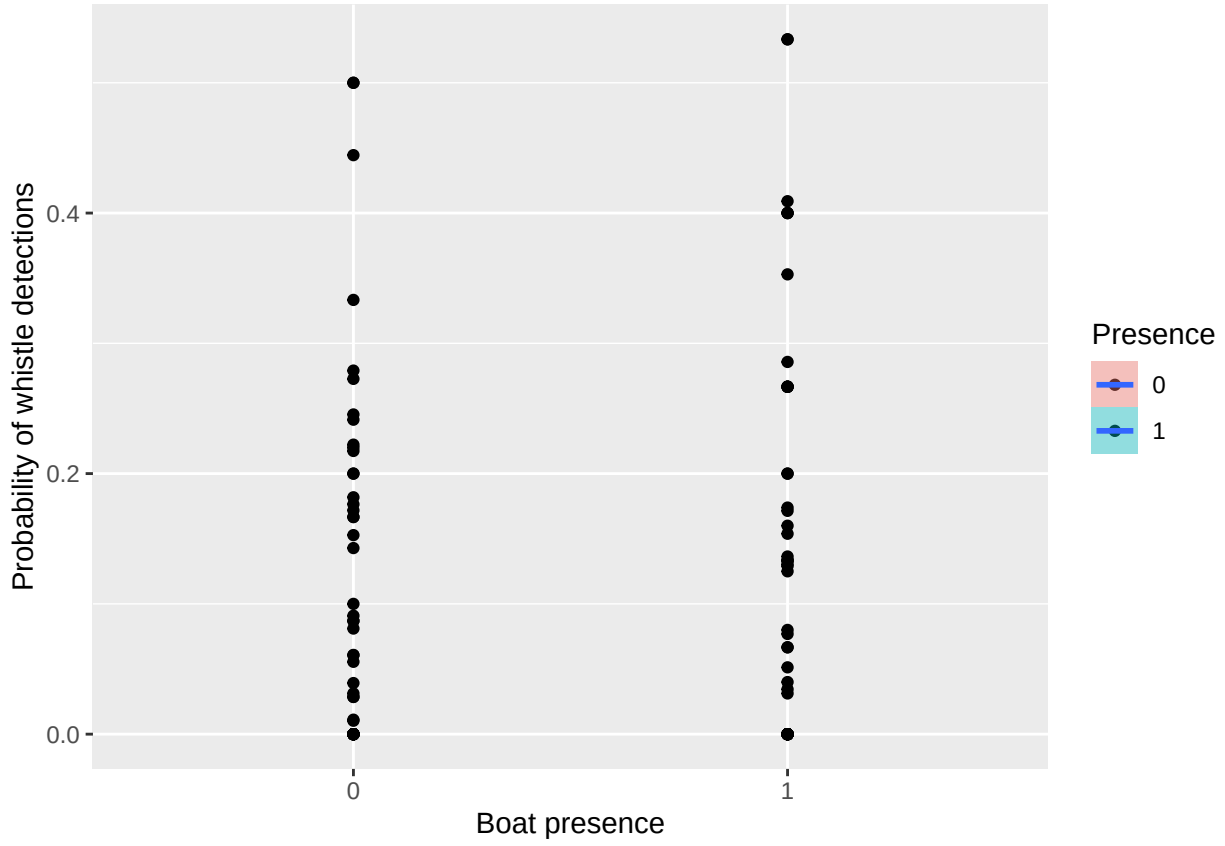
Boat

```
gg.boat.gam<-ggplot(boatboxdf, aes(x=Presence, y=Boat.prop, fill=Presence))+ geom_point() + geom_smooth
gg.boat.gam +labs(x="Boat presence", y="Probability of whistle detections")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 7 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 7 rows containing missing values (geom_point).
```




```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.0.5
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(gg.day.gam +labs(x="Time of day", y="Whistle detection probability"),
gg.tide.gam +labs(x="Time since high tide", y="Whistle detection probability"),
gg.boat.gam +labs(x="Boat presence", y="Whistle detection probability")+
    theme(legend.position="none"),ncol=2, nrow=2)
```

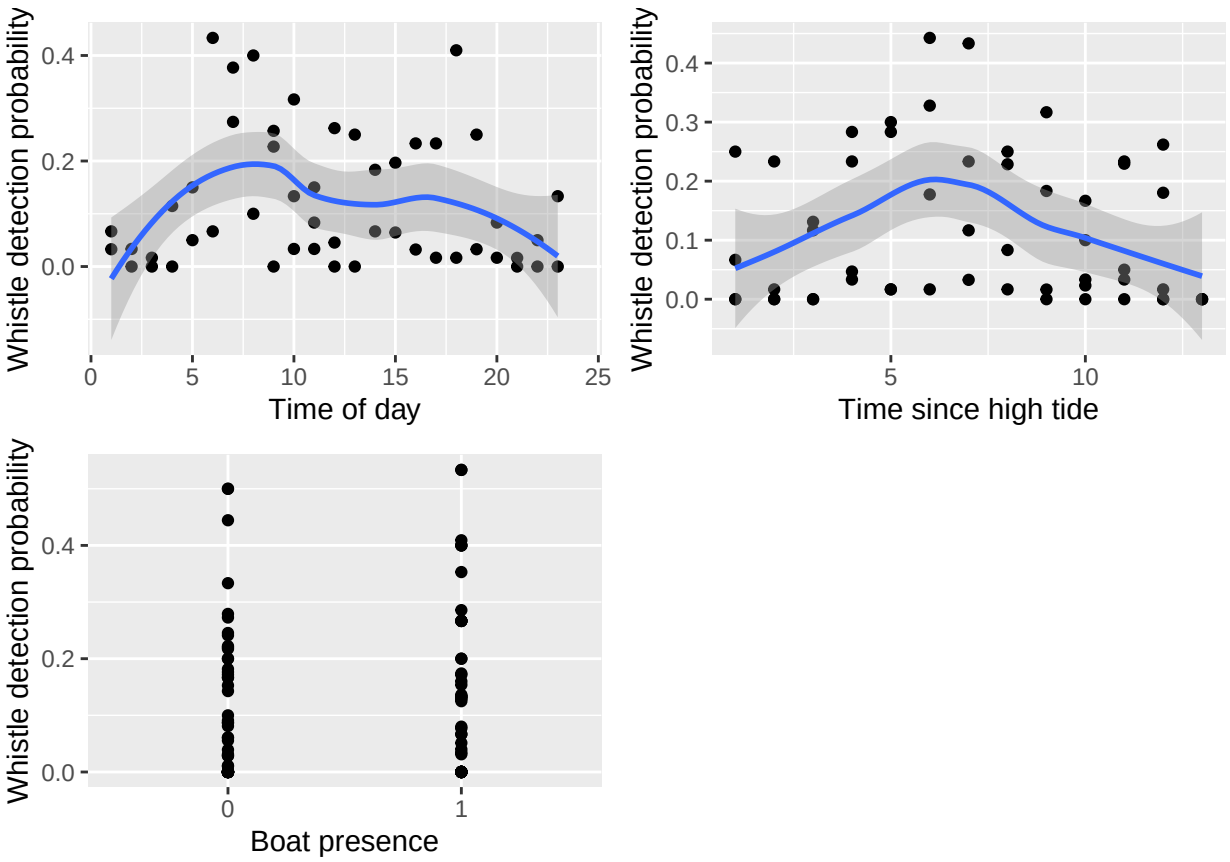
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 22 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 22 rows containing missing values (geom_point).
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 11 rows containing non-finite values (stat_smooth).

## Warning: Removed 11 rows containing missing values (geom_point).

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning: Removed 7 rows containing non-finite values (stat_smooth).

## Warning: Removed 7 rows containing missing values (geom_point).
```



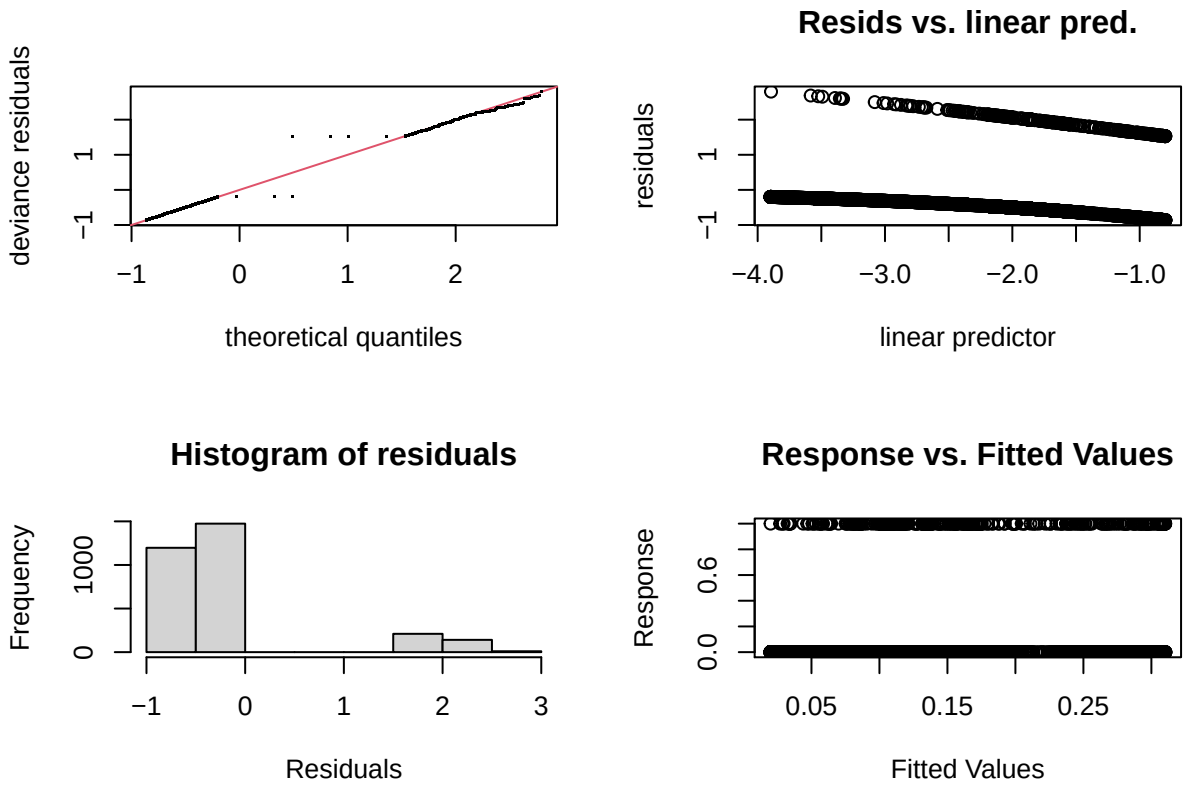
comparing deviance explained

```
summary.gam(gam1.3)
###radj:0.0559,dev explained:8.24, decTime most sig.
summary.gam(gam1.2)
###radj:0.0556,dev explained:8.22, decTime most sig.
summary.gam(gam1.1)
###radj:0.0245,dev explained:3.24, TSHT most sig.
summary.gam(gam1)
###radj:0.0558,dev explained:8.27, decTime most sig.

##Trying model with just diel cycle
```

```
gam2<- gam(whistle.pos.min~ + s(decTime ,bs="cc",k=24),family="binomial", data=finaldf.posixwhistles, m
summary(gam2)
###has much better diagnostics
```

Model selection

```
AIC(gam1,gam1.1,gam1.2,gam1.3,gam2)
###AIC lowest when only diel predictor is included in model(gam2)

###validating model using normalised residuals (before scaled residuals with DHARMa package)
par(mfrow= c(2,2))
gam.check(gam2)
```



using DHARMa package to test to see if family chosen in model is reasoning for residual patterns in model validation

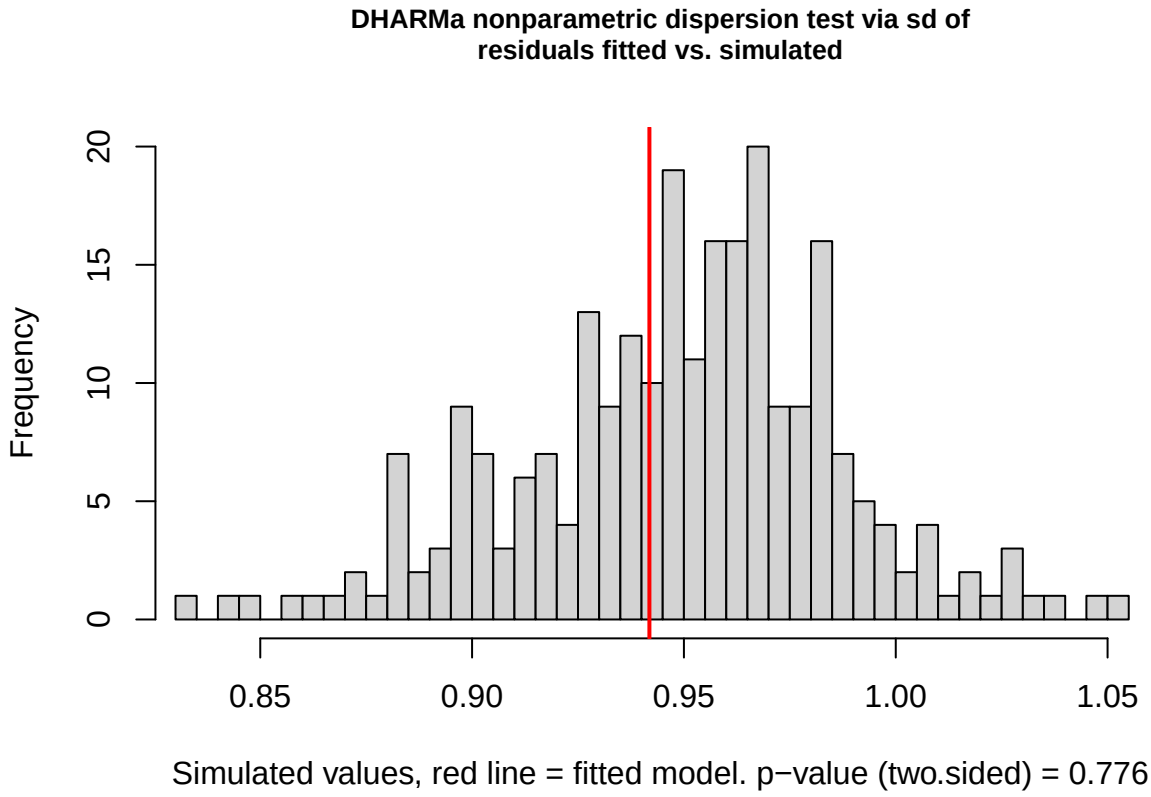
I first calculated the randomised quantile residuals of fitted model (GAM2)

```
library(DHARMa)
```

```
## Warning: package 'DHARMa' was built under R version 4.0.5
```

```
## This is DHARMa 0.4.3. For overview type '?DHARMa'. For recent changes, type news(package = 'DHARMa')
```

```
testDispersion(gam2)
```



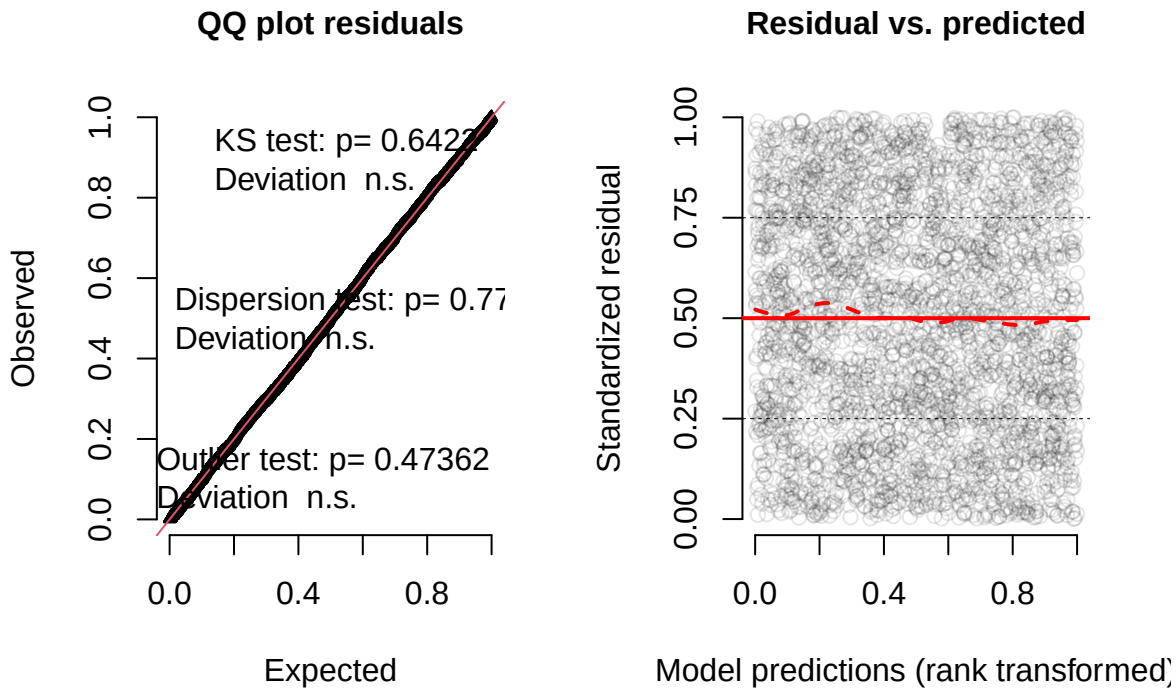

```
simulationOutput <- simulateResiduals(fittedModel = gam2, plot = F)
```

I then calculated its scaled residuals (mimicing straight line to see if residuals in model using binomial family are valid)

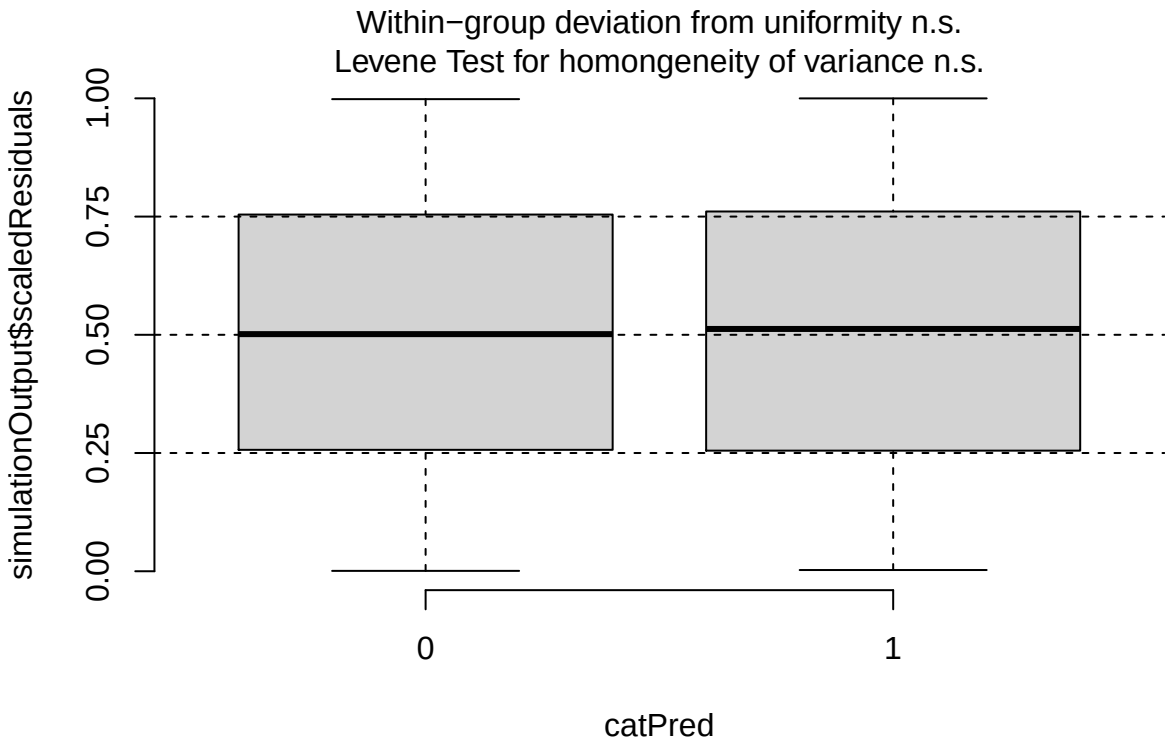
```
residuals(simulationOutput)
residuals(simulationOutput, quantileFunction = qnorm, outlierValues = c(-7,7))
```

I then tested to see if normality and homogeinity/dependance of residuals-checks out

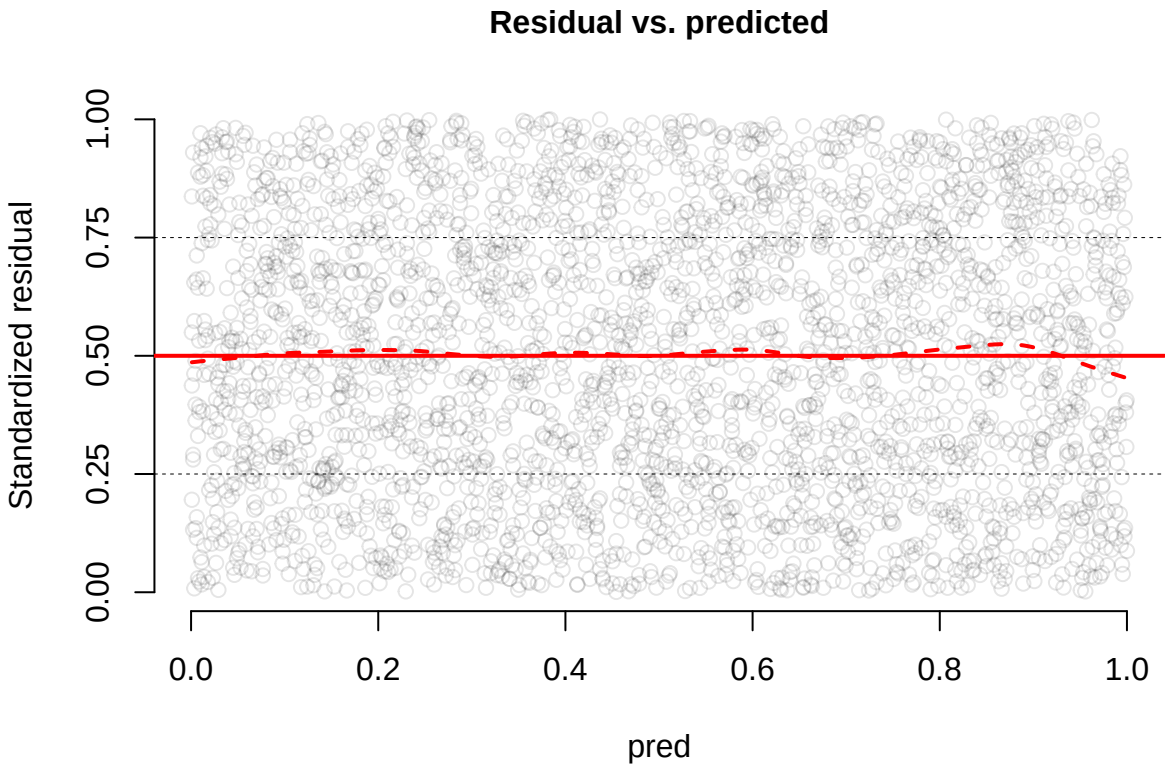
```
plot(simulationOutput)
```



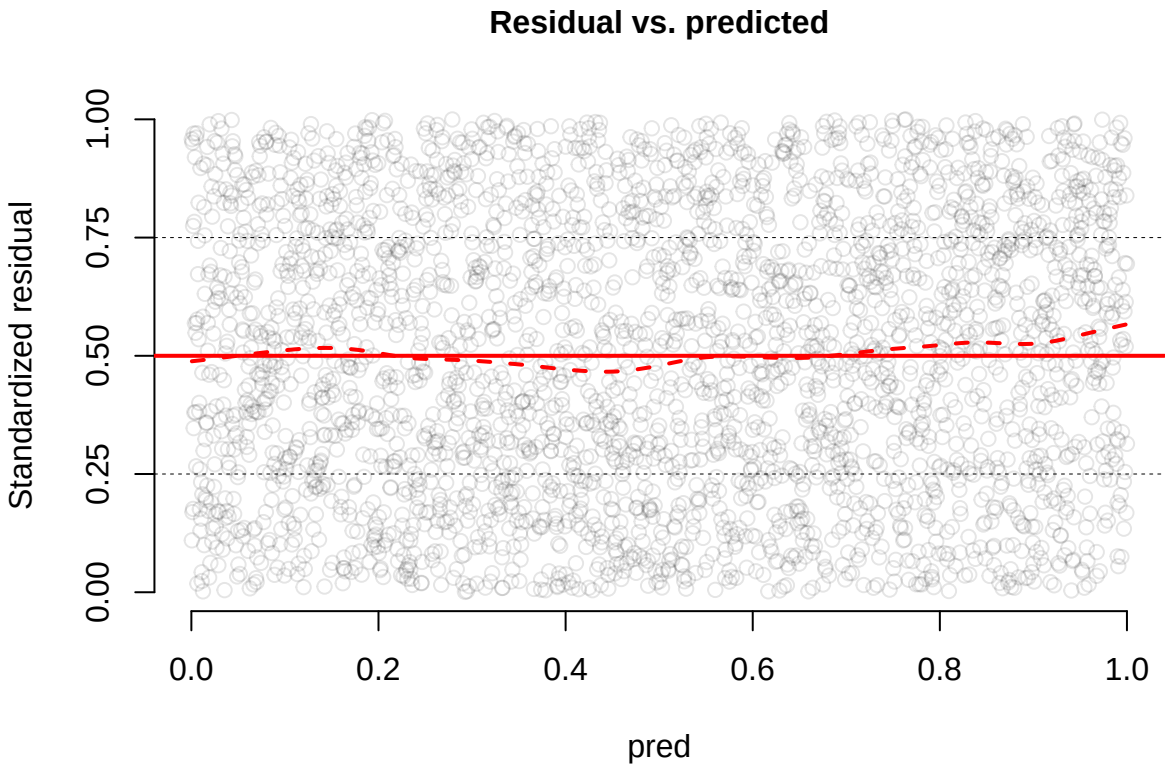

```
plotResiduals(simulationOutput, form = finaldf.posixwhistles$Fboatpresence)
```




```
plotResiduals(simulationOutput, form = finaldf.posixwhistles$TSHT)
```




```
plotResiduals(simulationOutput, form = finaldf.posixwhistles$decTime)
```




```
simulationOutput <- simulateResiduals(fittedModel = gam1, refit = F)
simulationOutput
```

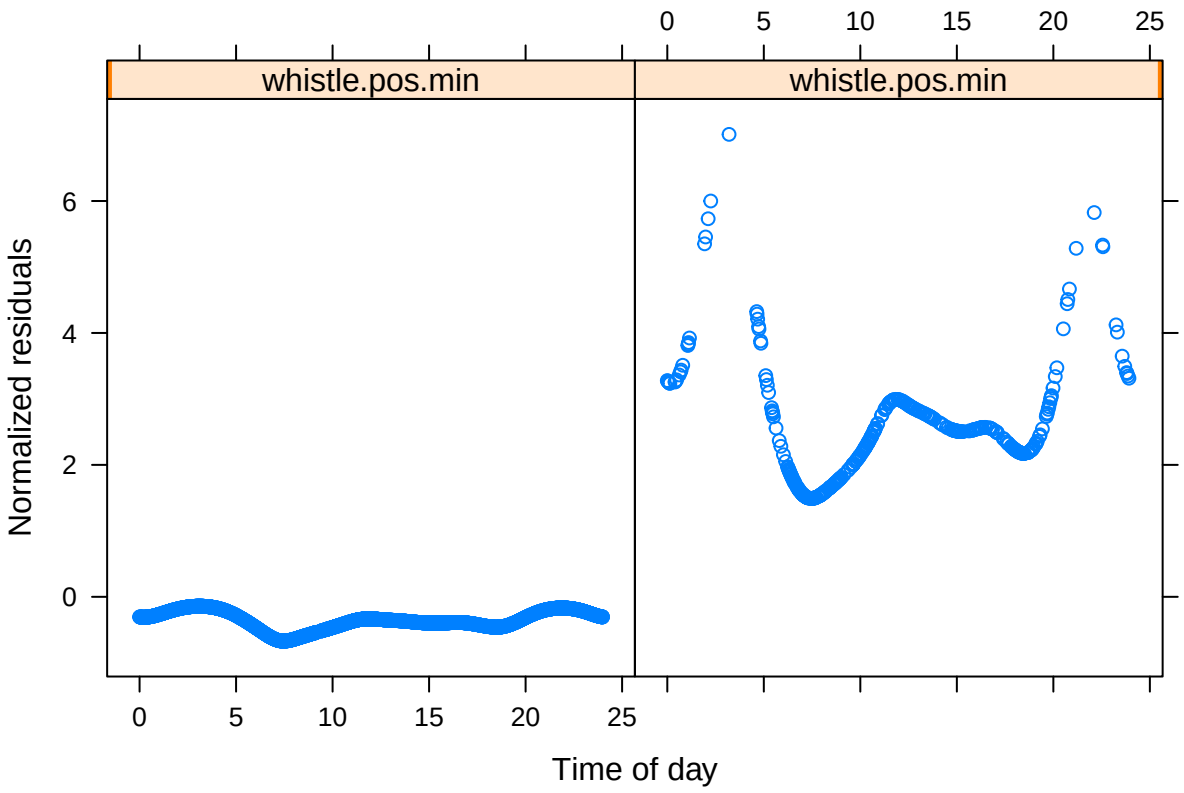
Checking if model residuals have temporal non-independence

To confirm temporal non-independence, extracted normalised residuals and plotted whistle proportion against each hour of diel cycle

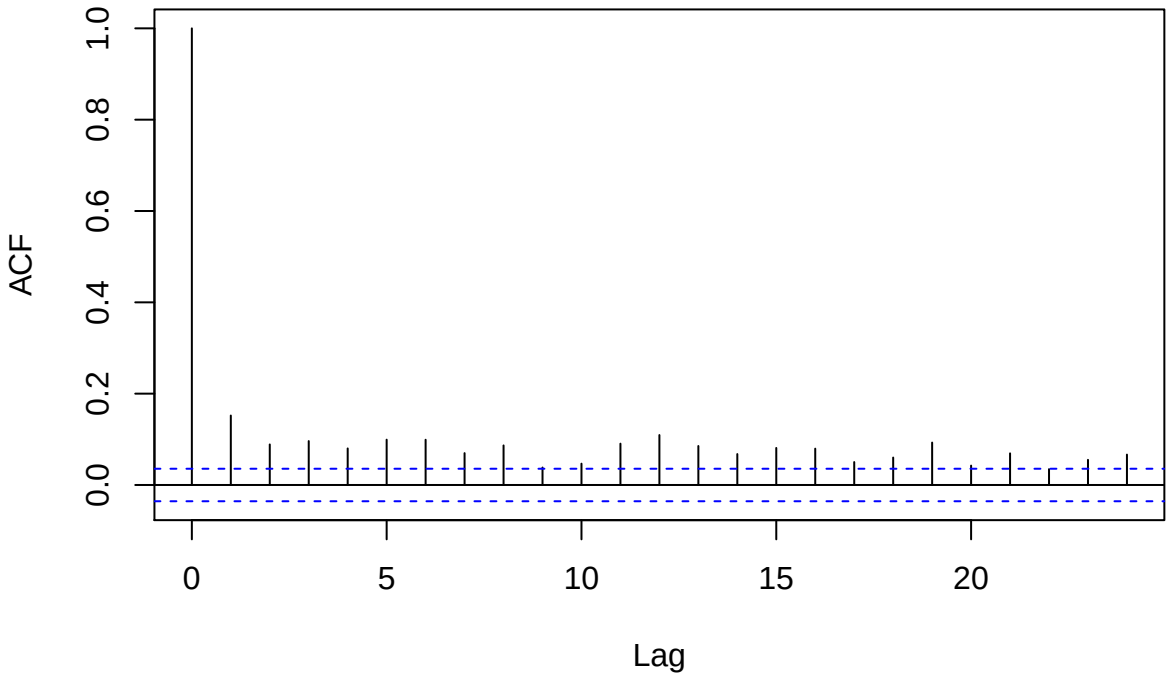
```
library(lattice)
```

```
## Warning: package 'lattice' was built under R version 4.0.5
```

```
xyplot(resid(gam2, type="pearson")~ decTime|whistle.pos.min, ylab="Normalized residuals", xlab="Time of
```

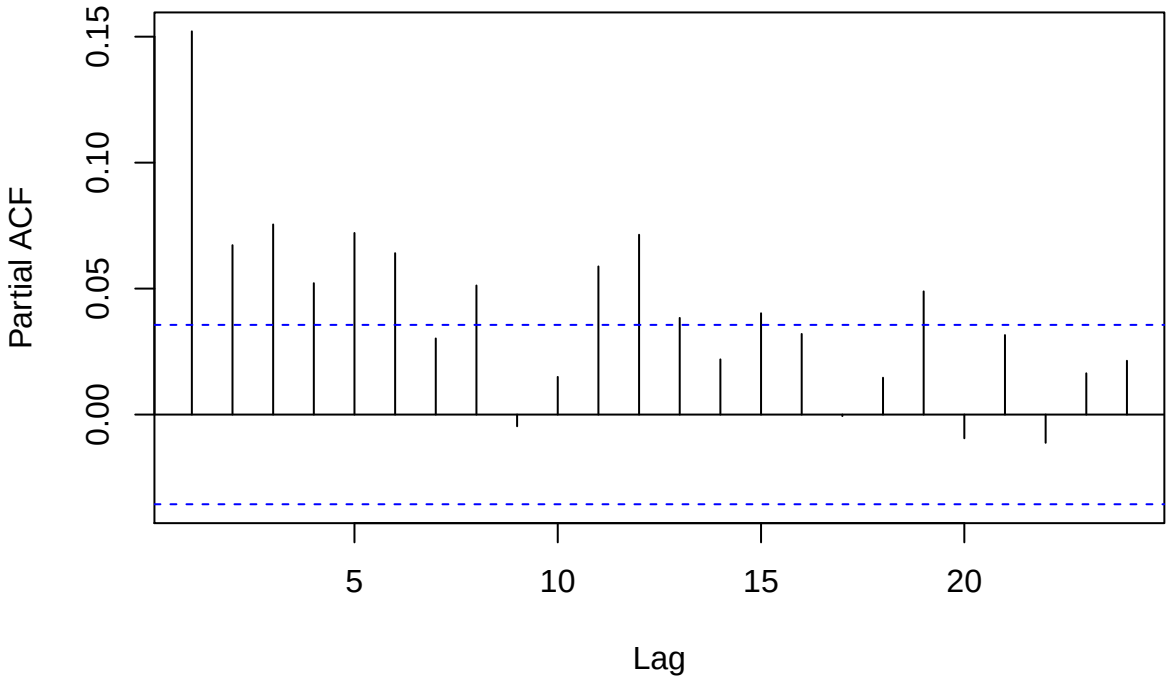



```
###used acf to decide which autocorrelation method could be used (would use MA-moving average if given
acf(residuals(gam1, type="pearson"),lag=24,main="")
```



Then used partial autocorrelation to calculate correlation of each lag but partials out/accounts for the correlation of previous lags, as more than 50% of lags are significantly different from zero (above— line), MA (moving average) structure is needed if had time to go ahead with fixing temporal non-independance

```
pacf(residuals(gam1, type="pearson"),lag=24,main="")
```


Partial ACF
0.00
0.05
0.10
0.15
5
10
15
20
Lag