Bioacoustics exploration using SilentCities data

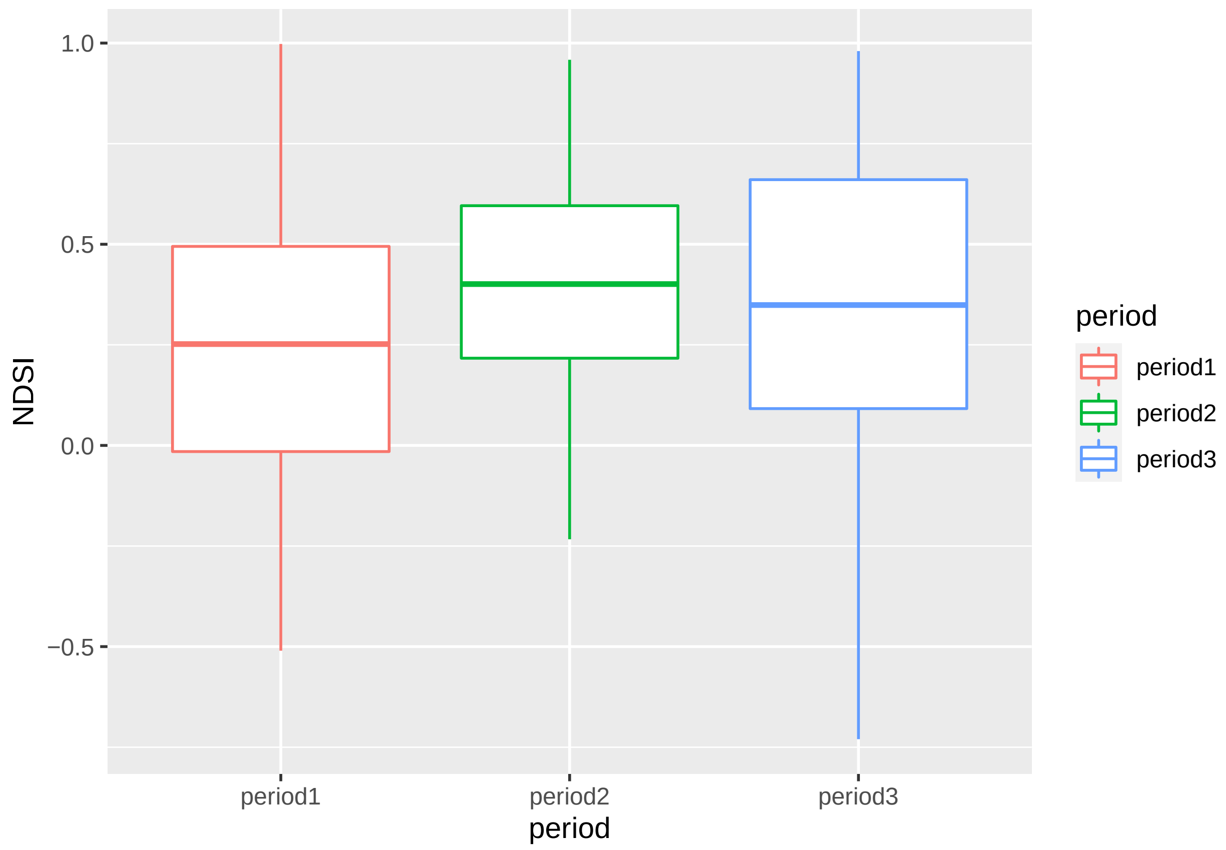
David Elliott

16 February 2021

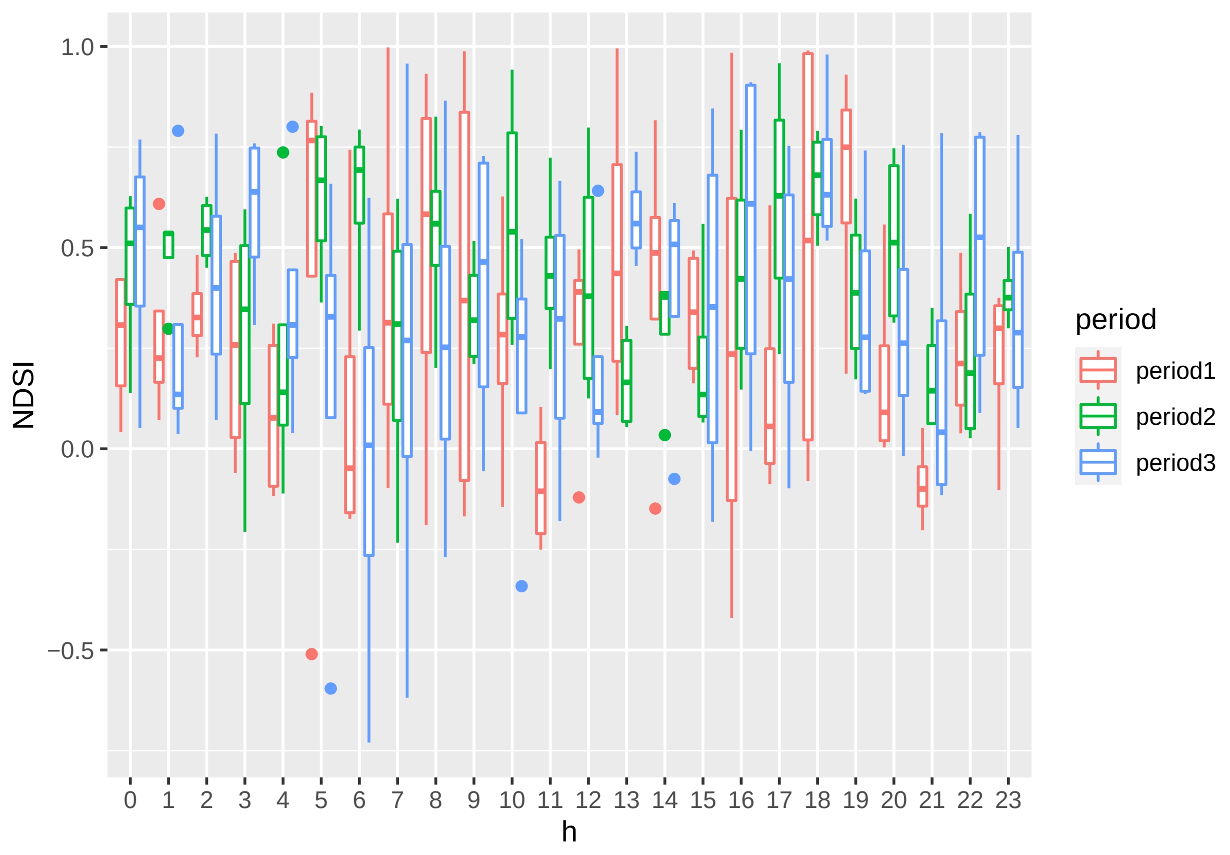
```r
library(seewave)
library(tuneR)
library(ggplot2)

# set up some variables
# start and end specified here are used in readWave
# For testing, use a small interval to keep the processing quick.
# Once all worked out, suggest to use the full 1 minute clips -> set end_s to 60
start_s <- 0
end_s <- 60

# by using a pattern, select the files corresponding to the beginning of each hour
# you could devise a pattern to select the files you want, or simply place the files
# into folders and make separate import commands per folder
audio.files <- list.files(full.names = "T",
                path = "O:\\sound\\SilentCities\\data-wav\\", pattern=".*0000.wav")

# on my system this found 4485 files
# for simplicity here I am selecting a small subset of them for demonstration purposes
period1.files <- audio.files[1:96]
period2.files <- audio.files[1101:1196]
period3.files <- audio.files[4001:4096]

# use the file lists to make three arrays of the wav file objects:

# initiate the list objects
audio1<-list()
audio2<-list()
audio3<-list()

# loop over the files and read them in using readwave
for(i in 1:length(period1.files))
  audio1[[i]]<-readWave(period1.files[i], from = start_s, to = end_s, units = "seconds")
for(i in 1:length(period2.files))
  audio2[[i]]<-readWave(period2.files[i], from = start_s, to = end_s, units = "seconds")
for(i in 1:length(period3.files))
  audio3[[i]]<-readWave(period3.files[i], from = start_s, to = end_s, units = "seconds")

# re-name the indexes of the array for ease of reference, to contain the date and time
# this would need customising for different participant IDs
names(audio1)<-gsub(".wav","",gsub("^.*OTHE0062_0\\+1_","",period1.files))
names(audio2)<-gsub(".wav","",gsub("^.*OTHE0062_0\\+1_","",period2.files))
names(audio3)<-gsub(".wav","",gsub("^.*OTHE0062_0\\+1_","",period3.files))

# join the lists together, because it is simpler later to analyse everything in one big loop
audio <- c(audio1,audio2,audio3)

# extract the hour from the filenames
hours <- as.integer(substr(names(audio),start = 10,stop = 11))

# set up a data frame to hold the analysis results
audio.analysis <- data.frame(matrix(ncol=7,nrow=length(audio),
                dimnames=list(NULL,
                c("period","hour","file","NDSI","anthrophony","biophony","entropy"))))
audio.analysis[1:96,"period"] <- "period1"
audio.analysis[97:192,"period"] <- "period2"
audio.analysis[193:288,"period"] <- "period3"
audio.analysis$period <- as.factor(audio.analysis$period)
audio.analysis$hour <- hours

# analyse by running a loop over all the files, and store results in the dataframe made above
for(i in 1:length(audio)){
  audio.soundscape <- soundscapespec(audio[[i]],plot="FALSE")
  audio.analysis[i,"file"] <- names(audio[i])
  audio.analysis[i,"NDSI"] <- NDSI(audio.soundscape, anthropophony = 1, biophony = 2:8, max = FALSE)
  audio.analysis[i,"anthrophony"] <- mean(audio.soundscape[1:2,])
  audio.analysis[i,"biophony"] <- mean(audio.soundscape[3:8,])
  audio.analysis[i,"entropy"] <- H(audio[[i]])
}

# the following command stores the hour as a factor called "h"
# this is to give more plotting options
audio.analysis$h <- as.factor(audio.analysis$hour)

# results are stored in the dataframe audio.analysis
# the analysis commands are from the seewave package
# It is worth reading the documentation from the commands, and the referenced papers

# anthrophony and biophony were calculated from the amplitude of binned frequencies,
# of the soundscapespec command
# anthropy is the average amplitude below 2000 hz
# biophony is the average amplitude between 2000 - 8000 hz
# To understand this, try running the soundscapespec command on any audio file
# and also read the documentation.
# However, I don't think you really need this because the NDSI function calculates
# exactly what you were trying to get.
# Please read the docs for that.
# I suggest also the following 2 papers:
# http://www.iinsteco.org/people/pubblications/almo/2014/2014_Acoustic_indices_for_biodiversity_assessm
# https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0004065


# Plot some results from the analysis
# (focusing here on NDSI: higher values have proportionally more anthrophony)
# Note that these are generally not valid analyses because there was not
# much thought given to the data selection
# in particular, it is important to consider what is the unit of replication
# (in this case it was
end_s - start_s
```

```
## [1] 60
```

```r
# seconds of each hour.
# Probably you should select a specific time or even better take whole days average)
# This is just for demonstration how to do the analysis and plotting.
# You should be able to expand on it to make it more robust

ggplot(data=audio.analysis,aes(x=period,y=NDSI, colour=period)) + geom_boxplot()
```

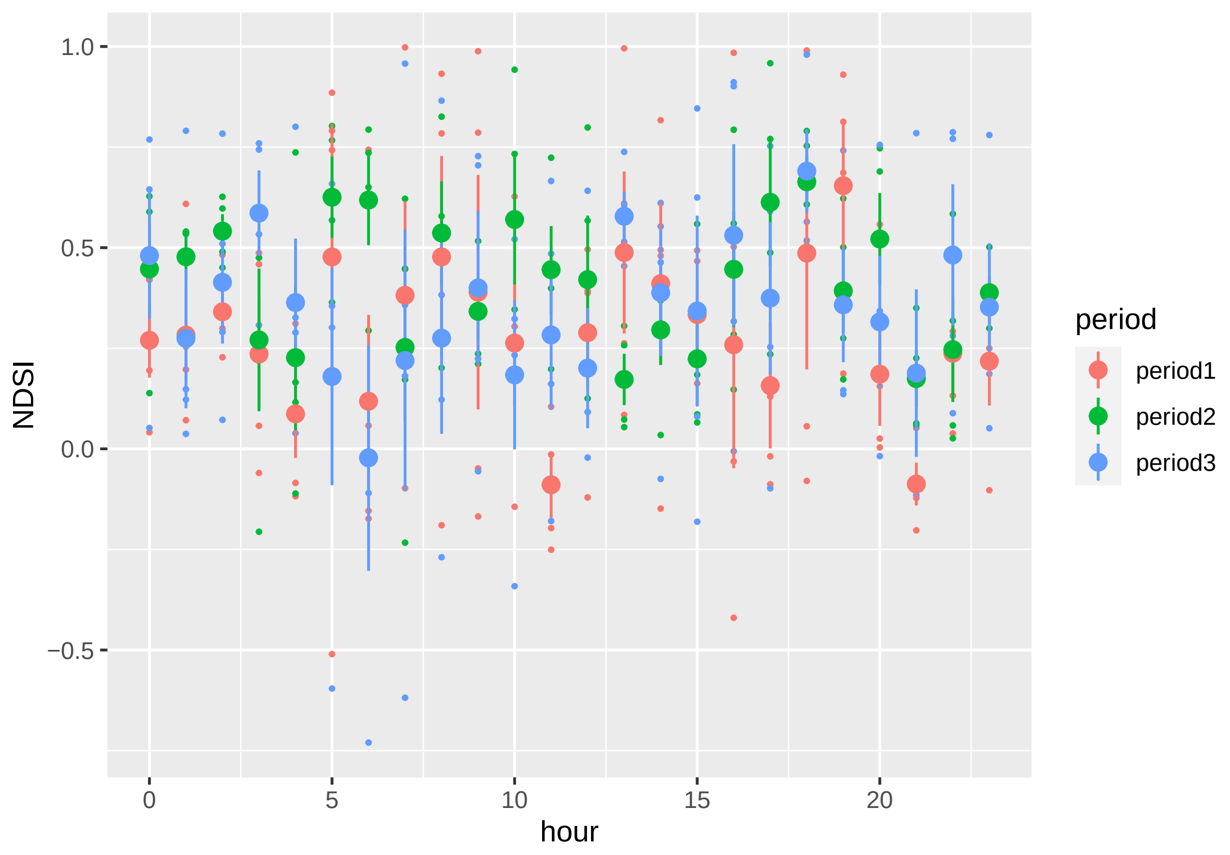


```r
ggplot(data=audio.analysis,aes(x=h,y=NDSI, colour=period)) + geom_boxplot()
```

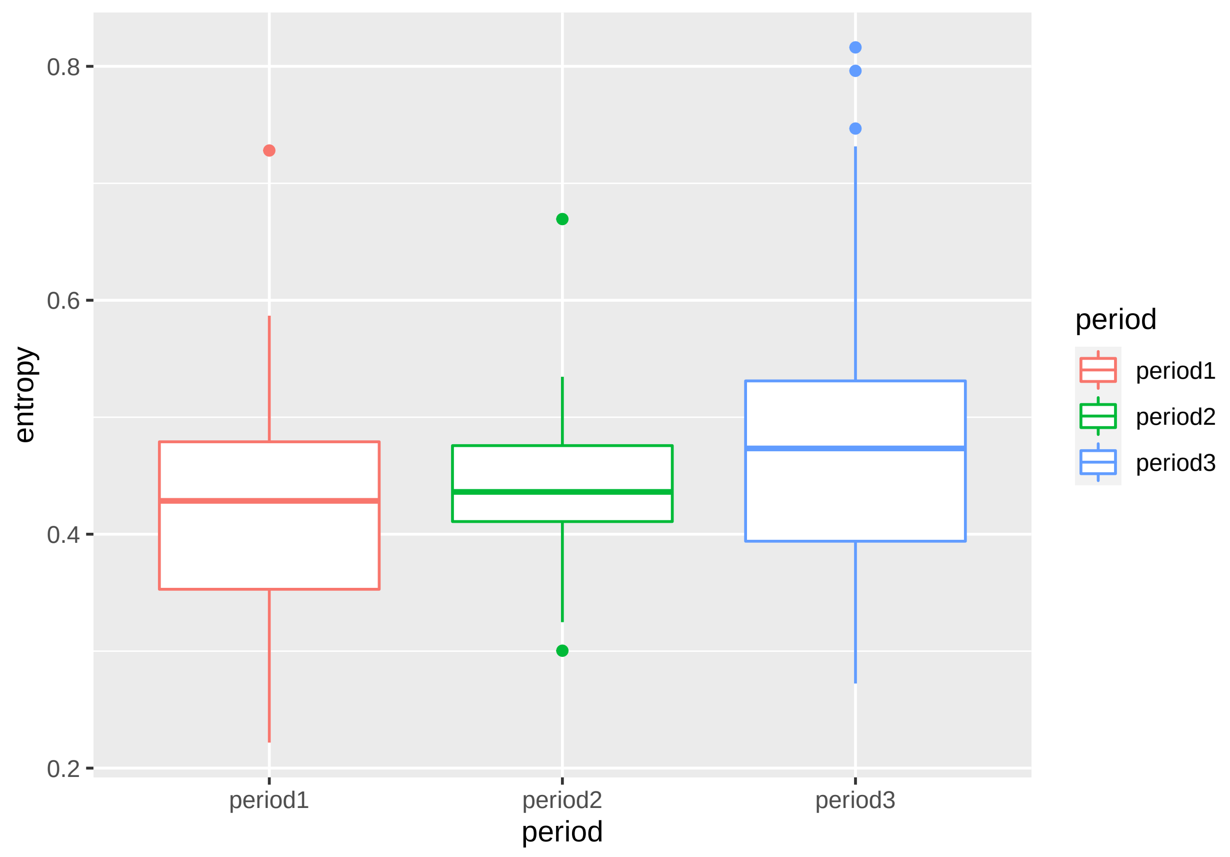


```r
ggplot(data=audio.analysis,aes(x=hour,y=NDSI, colour=period)) + geom_point(size=0.5) + stat_summary()
```

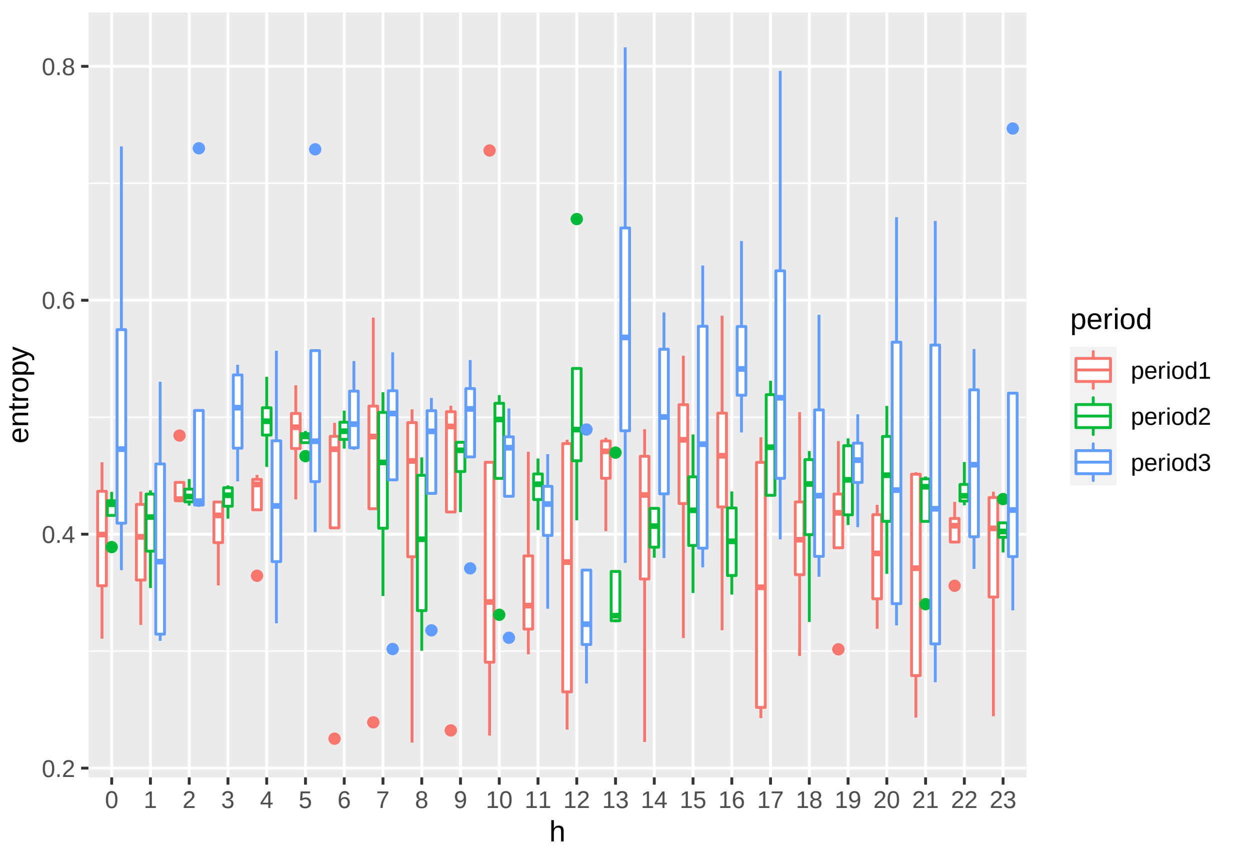
```
## No summary function supplied, defaulting to `mean_se()`
```



```r
# some similar plots but this time looking at entropy
ggplot(data=audio.analysis,aes(x=period,y=entropy, colour=period)) + geom_boxplot()
```

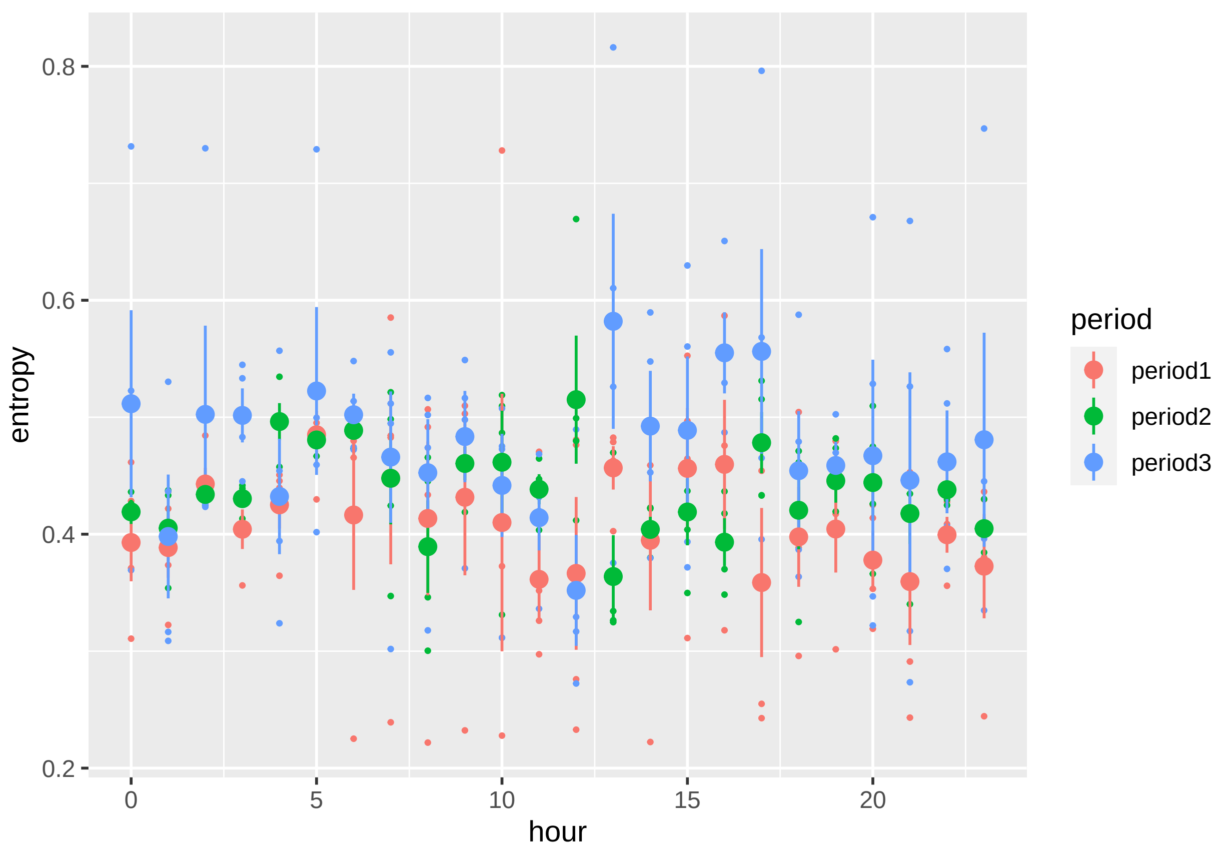


```r
ggplot(data=audio.analysis,aes(x=h,y=entropy, colour=period)) + geom_boxplot()
```



```
ggplot(data=audio.analysis,aes(x=hour,y=entropy, colour=period)) + geom_point(size=0.5) + stat_summary(
```

```
## No summary function supplied, defaulting to `mean_se()`
```


0.8
0.6
0.4
0.2
entropy
0
5
10
15
20
hour
period
period1
period2
period3