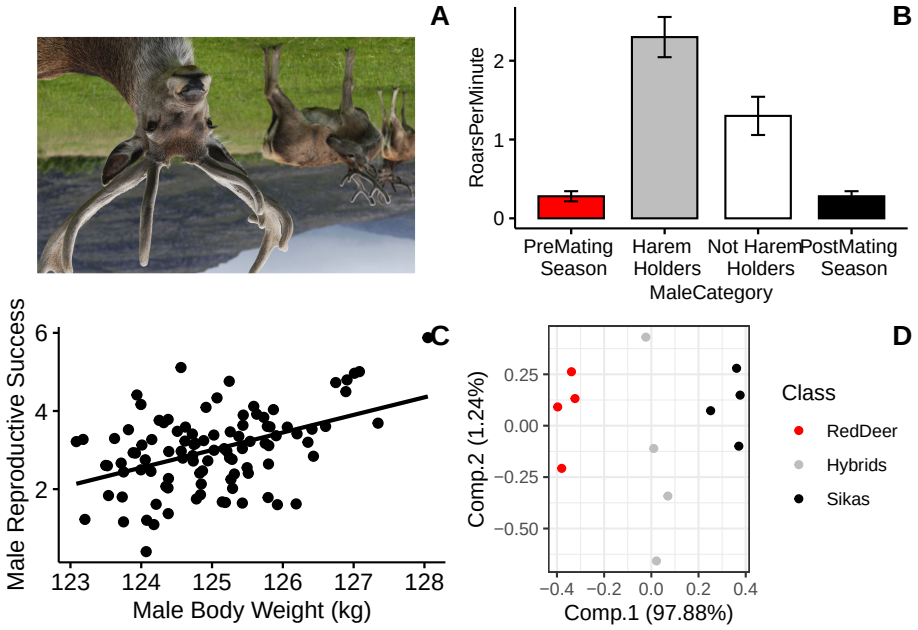
behaviouR: R package and tutorials for online teaching of fundamental concepts in behavior and ecology

Dena J. Clink

2020-08-04

Contents

Welcome



To get started you should download the package from Github using the following code.

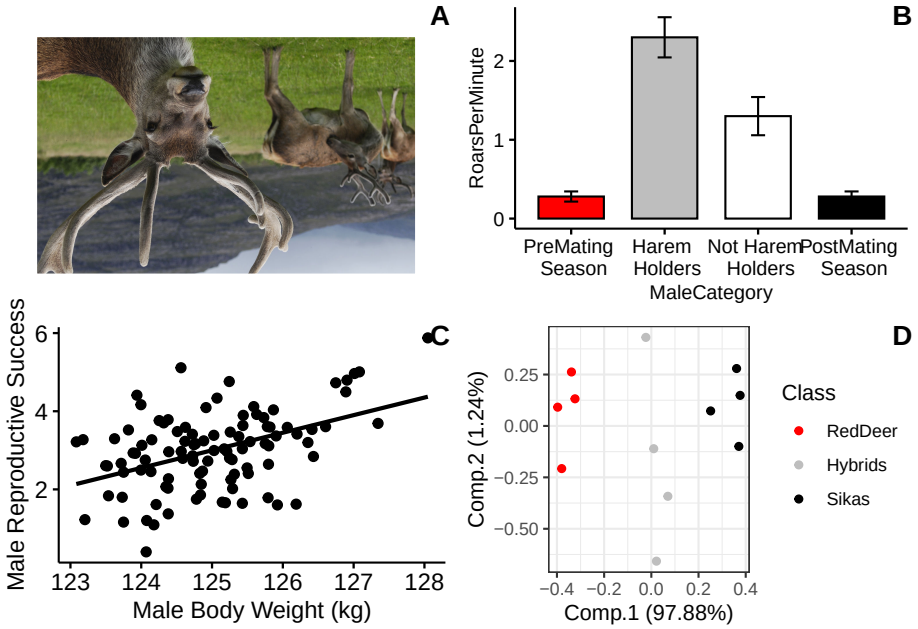
```
# Install devtools from CRAN
install.packages("devtools")
devtools::install_github("https://github.com/DenaJGibbon/behaviouR")
```

Note: If you have never used R before I highly recommend that you check out the primers here: https://rstudio.cloud/learn/primers.

You can navigate using the tabs at the left and/or the arrows.

Chapter 1

Lab 1. Data exploration and visualization



Background
Our examples will focus on roaring behavior in red deer. Red deer on Rum Island, Scotland form harems during the mating season. Harem holding males fight off other males during this time, and many of these males have signs of fighting injuries. Clutton-Brock and colleagues hypothesized that because of the

injury associated with fights that red deer males should use honest indicators of each others fighting abilities, and that the main indicator of strength and fighting ability was the male roars.

For more background see: Clutton-Brock, Tim H., and Steven D. Albon. "The roaring of red deer and the evolution of honest advertisement." *Behaviour* 69.3-4 (1979): 145-170.

Reby D, McComb K. Anatomical constraints generate honesty: acoustic cues to age and weight in the roars of red deer stags. Animal behaviour. 2003 Mar 1;65(3):519-30.

Long, A. M., N. P. Moore, and T. J. Hayden. "Vocalizations in red deer (*Cervus elaphus*), sika deer (*Cervus nippon*), and red× sika hybrids." Journal of Zoology 244.1 (1998): 123-134.

Goals of the exercises
The main goal(s) of today's lab are to:
1) teach you about different types of data visualization and types of data.
2) help you to start to become familiar with using R.
3) to get you to think about the ways that scientists analyze data.

Getting started
First we need to load the relevant package. Packages contain all the functions that are needed for data analysis. The 'behaviouR' package loads many other packages that have important functions.

```
# First we load the relevant packages
library(behaviouR)
```

Lab 1a. Categorical data

Categorical variables represent types of data that can be divided into groups. Our first example will census a simulated population of red deer to determine the proportion of individuals in different developmental stages.

```
## Lab 1a. Categorical data
# Here we create a simulated population with four categories (Infant, Juvenile, AdultF
DeerPopulationDF <- data.frame(DevelopmentStage=c('Infant','Juvenile','AdultFemale','Ad
                               NumberOfIndividuals=c(15,50,125,200))

# We then print the object so that we can see the output
DeerPopulationDF
```

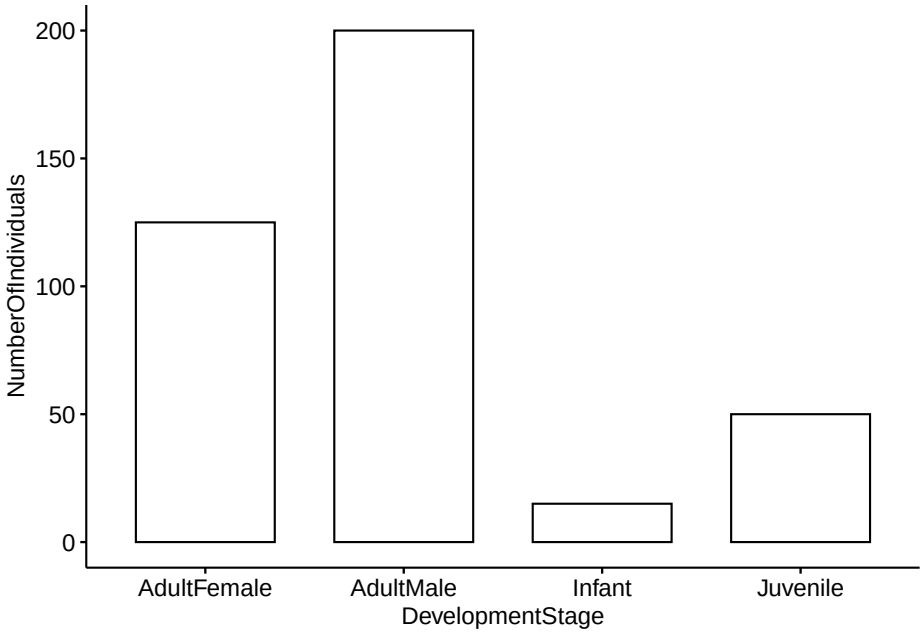
```
##   DevelopmentStage NumberOfIndividuals
## 1           Infant                  15
## 2         Juvenile                  50
## 3      AdultFemale                 125
```

```
## 4        AdultMale                  200
```

Now we want to plot the data. We will use a simple barplot to start.

Throughout these tutorials we will use functions from the 'ggpubr' package for plotting; see https://rpkgs.datanovia.com/ggpubr/ for more details. For more information on the great work the developer is doing see http://www.alboukadel.com/ and http://www.sthda.com/english/.

```
# Now we want to plot the data. We will use a simple barplot to start.
ggbarplot(DeerPopulationDF, x='DevelopmentStage', y='NumberOfIndividuals')
```



Please answer the following questions:
Question 1. How would you interpret this figure? Which category has the most individuals, and which has the least?

Question 2a. This is a simulated popuation, but what do you think the ratio of males to females would mean for male-male competition?

Question 2b. Modify the code above so that our simulated deer population has a more even ratio of males to females.

Lab 1b. Categorical and continuous data

Our second example will investigate roaring rates in deer as a function of status and breeding season.The categories we will use are: pre-mating season, harem holder, not harem holder and post-mating season.Our outcome variable (mean

number of roars per minute) is continuous; it represents actual numerical measurements.

I created a toy dataset which is based on Figure 2 in Clutton-Brock et al. 1979 where for each category we have five observations which represent the average number of calls per minute. For each category we have a total of five observations.

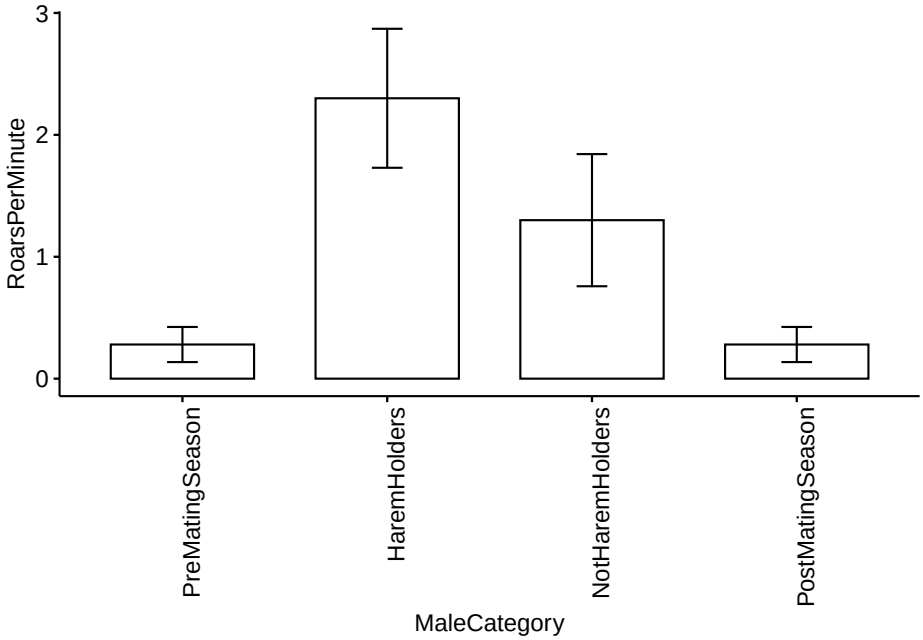
```
## Lab 1b. Categorical and continuous data
# Load the dataset so that we can use it
data("MaleDeerRoarDF")

# We can check the structure of the dataframe by using the command 'head'
head(MaleDeerRoarDF)
```

```
##      MaleCategory RoarsPerMinute
## 1 PreMatingSeason           0.25
## 2 PreMatingSeason           0.50
## 3 PreMatingSeason           0.25
## 4 PreMatingSeason           0.30
## 5 PreMatingSeason           0.10
## 6    HaremHolders           2.50
```

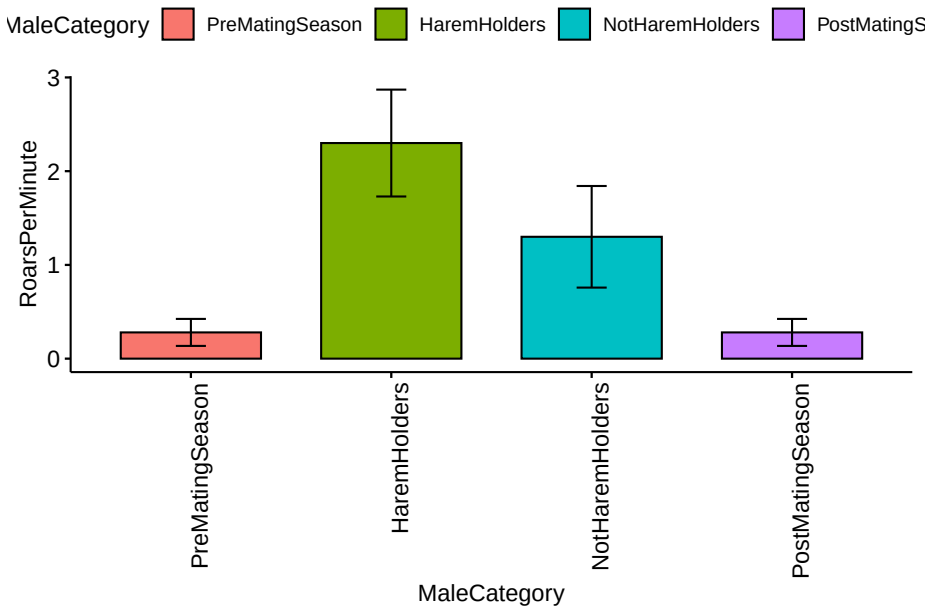
Now we will plot the categorical data. First we will use a barplot with error bars representing the standard deviation. See https://www.biologyforlife.com/interpreting-error-bars.html and https://www.biologyforlife.com/standard-deviation.html for more information about error bars.

```
# Now we will plot the categorical data with standard deviations.
ggbarplot(MaleDeerRoarDF, x='MaleCategory', y='RoarsPerMinute',
          add = c("mean_sd"),xtickslab.rt = 90)
```



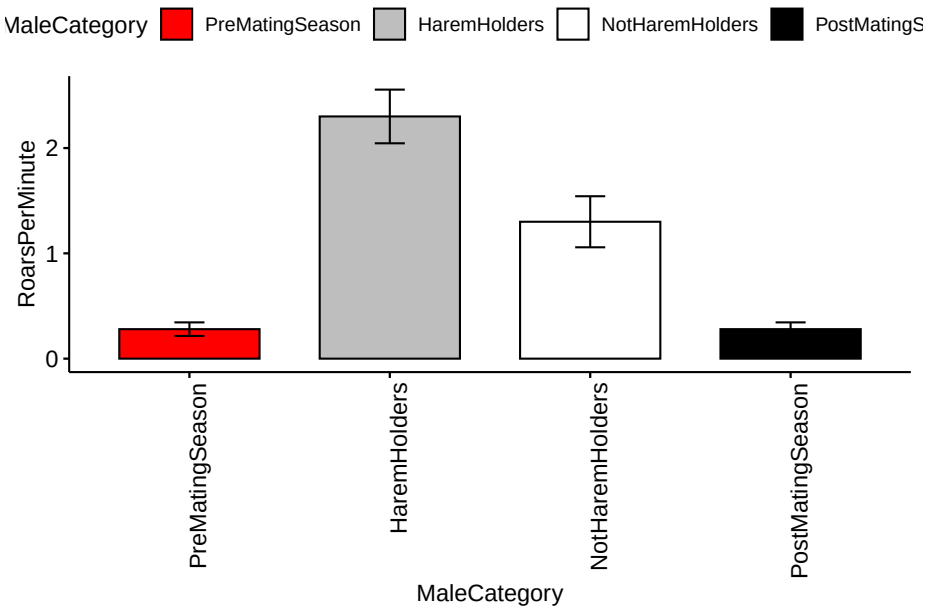
Although it shows us what we need, we can also include some colors in our plot. To do this I added the 'fill = 'MaleCategory' argument, which tells R to color the plot based on male categories.

```
# Now we will plot the categorical data with colors for each category.
ggbarplot(MaleDeerRoarDF, x='MaleCategory', y='RoarsPerMinute', fill = 'MaleCategory',
          add = c("mean_sd"),xtickslab.rt = 90)
```



Color for plots is often based on personal preference, below I modified the colors using the 'palette' argument. NOTE: The use of color-blind friendly palettes is becoming increasingly popular among scientists.

```
# Now we will plot the categorical data with user-specified colors for each category.
ggbarplot(MaleDeerRoarDF, x='MaleCategory', y='RoarsPerMinute', fill = 'MaleCategory',
          palette = c('red','gray','white','black'),
          add = c("mean_se"),xtickslab.rt = 90)
```



Please answer the following questions:

Question 3. How do you interpret the barplot figure?

Question 4. Visit this site (http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf) and change the colors of the plot. Hint: change 'palette = c('red','gray','white','black')'to 'palette = c('color1','color2','color3','color4')'.

Lab 1c. Categorical and continuous data

Continuous data represent numerical measurements, and when both our variables of interest are continuous we can plot them in a different way.

For this example we will consider the relationship between male red deer body weight (in kilograms) and male reproductive success (which is estimated by the number of days that he associated with females during the breeding season).

Below we have a function that will simulate data for us, and we can specify the level of correlation.

We also assign a mean body weight for males in our population (I got this value from). We can also change the male reproductive success value. Our starting correlation value between these two variables is that reported in Reby & McComb 2003.

```
## Lab 1c. Categorical and continuous data
# This is the function that simulates our data. N is the number of individuals,
```

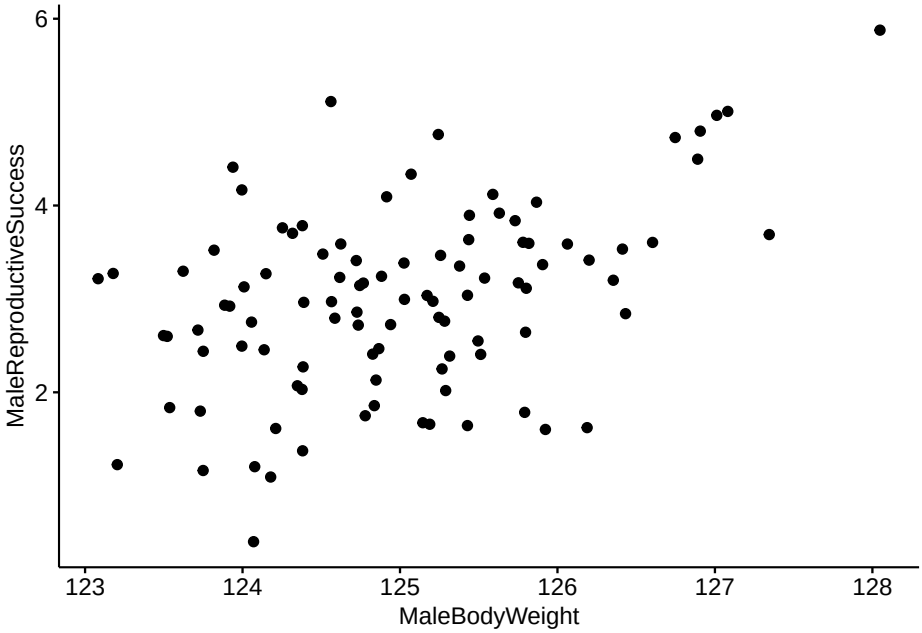
```
# CorrelationCoefficient tells us how correlated our data are,
# MaleMeanBodyWeight is the mean body weight of males in our population and
# MeanFemalesInHarem is the mean number of females in the harem.
MaleRedDeerDF <- CorrelatedDataSimulationFunction(N=100,
                                                  CorrelationCoefficient= 0.45,
                                                  MaleMeanBodyWeight = 125,
                                                  MaleReproductiveSuccess = 3)

# We can check the output
head(MaleRedDeerDF)
```

```
##   MaleBodyWeight MaleReproductiveSuccess
## 1       125.2576                3.466009
## 2       125.1888                1.658102
## 3       124.7266                2.858253
## 4       124.1372                2.455370
## 5       125.4283                1.643711
## 6       124.3844                2.272255
```

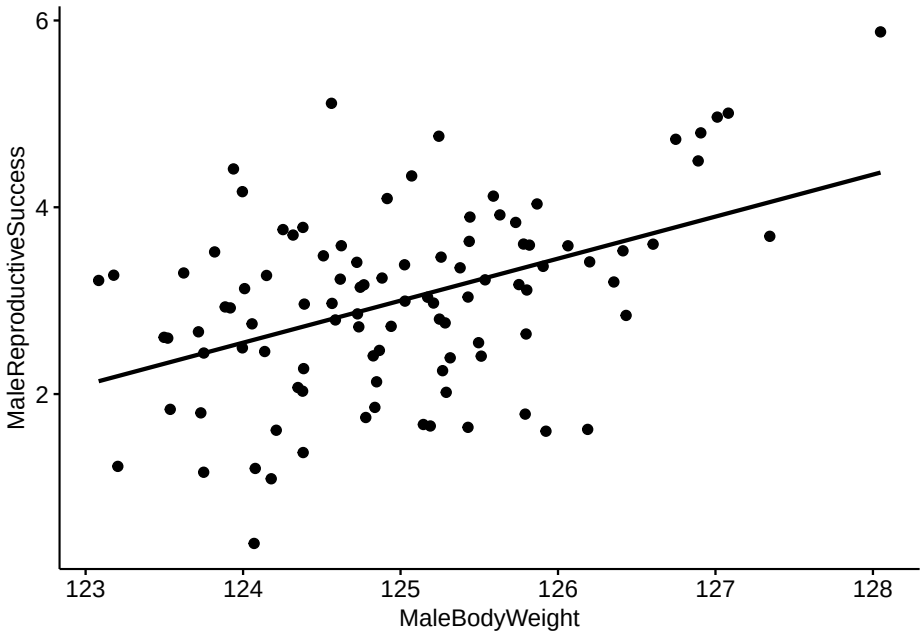
Now we plot the data the data as before, but this time we make a scatterplot

```
# Make a scatterplot of the data.
ggscatter(data=MaleRedDeerDF,x='MaleBodyWeight',y='MaleReproductiveSuccess')
```



It looks like there is a correlation between our two variables, but it would be better visualized with a trend line.

```
# Make a scatterplot with a trendline.
ggscatter(data=MaleRedDeerDF,x='MaleBodyWeight',y='MaleReproductiveSuccess',
          add='reg.line')
```



Now we can use R to create a linear model to quantitatively investigate the relationship between our two variables

```
# Create a linear model where MaleBodyWeight is the independent variable
# and FemalesInHarem is the dependent variable.
MaleDeerModel <- lm(MaleReproductiveSuccess ~ MaleBodyWeight,data=MaleRedDeerDF)
```

We can look at the output of the model and we see that our coefficient for MaleBodyWeight is 0.83. We interpret this to mean that for every 1 kg increase in male body mass, we would expect to see a 0.83 increase in the number of females in his harem.

```
# We can look at the output of the model
MaleDeerModel
```

```
##
## Call:
## lm(formula = MaleReproductiveSuccess ~ MaleBodyWeight, data = MaleRedDeerDF)
##
## Coefficients:
##    (Intercept)  MaleBodyWeight
##         -53.25            0.45
```

There are many ways that we can test whether the results of our model are reliable. A common way is through the use of p-values; a somewhat more intuitive way is through the use of model comparison using Akaike information criterion (AIC). AIC provides an estimate of how well the model fits the data.

NOTE: AIC can be used to compare two our more models created using the same dataset, but the relative AIC values are what is important, and you must use the same data for all models.

We will set up two models for this exercise. Our first model will be our 'null'model which does not include MaleBodyWeight as a predictor. Our next model will be our model of interest that does contain MaleBodyWeight as a predictor.

```
# Create a null model and a model with MaleBodyWeight as a predictor.
MaleDeerNull <- lm(MaleReproductiveSuccess ~ 1, data=MaleRedDeerDF)
MaleDeerModel <- lm(MaleReproductiveSuccess ~ MaleBodyWeight,data=MaleRedDeerDF)
```

We can now use a function created to compare models using a modified version of AIC (adjusted for small sample sizes).

```
# Compare models using AIC.
bbmle::AICctab(MaleDeerModel,MaleDeerNull, weights=T)
```

```
##                dAICc df weight
## MaleDeerModel   0.0  3  1
## MaleDeerNull   20.5  2  <0.001
```

NOTE: In this example case the model which contains MaleBodyWeight is ranked higher. We interpret this to mean that there is a reliably positive relationship between MaleBodyWeight and MaleReproductiveSuccess. In other words, as male body weight increases we see an increase in his reproductive success.

Please answer the following questions:

Question 5a. What happens when you change the correlation coefficent from 0.45 to a much smaller number and re-run the code?

Question 5b. What about when you change it to a much bigger number?

Lab 1d. Multivariate data

In some cases we have multiple measurements of different variables from the same individual; in this case our data would be considered 'multivariate'.

Lets create a toy dataset of red deer, sika deer and red x sika hybrid vocalizations (based on Long et al. 1998). We will simulate four vocalizations per individual and we will measure three aspects of the vocalizations- the duration,

the minimum frequency and the maximum frequency of the vocalization. We will then visualize our data using principal component analysis (PCA). PCA is a commonly used data reduction technique.

For more information see https://www.nature.com/articles/nmeth.4346.pdf.

```
## Lab 1d. Multivariate data
# Just as before we load our data
data("DeerSpeciesAcousticFeatures")

# Check the structure
head(DeerSpeciesAcousticFeatures)
```

```
##   Duration MinFrequency MaxFrequency   Class
## 1       15           50          125 RedDeer
## 2       14           49          127 RedDeer
## 3       12           51          126 RedDeer
## 4       13           52          127 RedDeer
## 5       17           35          125 Hybrids
## 6       19           37          126 Hybrids
```

Now we run the PCA. Note: You can only do PCA on numeric data, so we remove the class category.

```
# Here is our modified dataset that we will use for PCA
DeerSpeciesAcousticFeatures[,-c(4)]
```

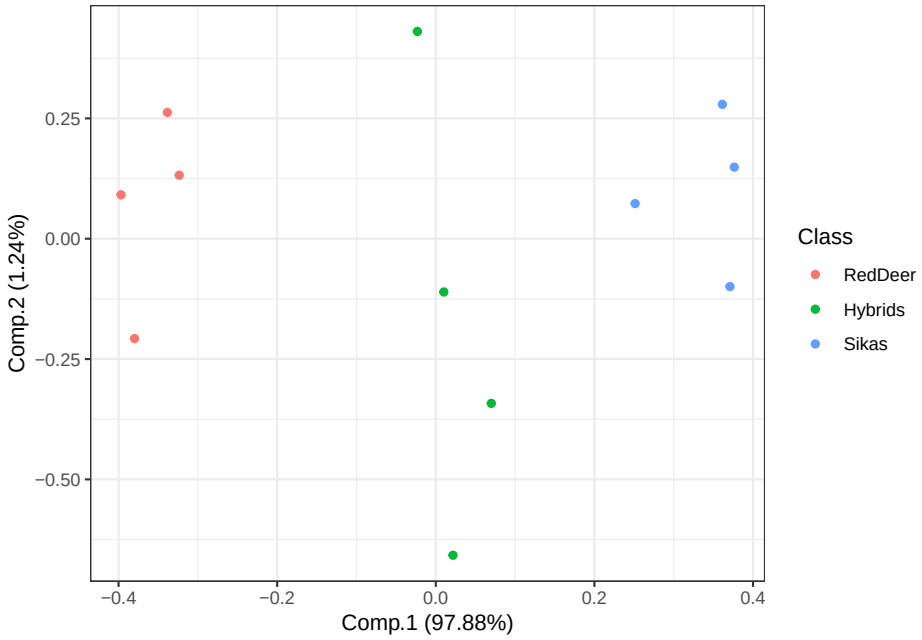
```
##    Duration MinFrequency MaxFrequency
## 1        15           50          125
## 2        14           49          127
## 3        12           51          126
## 4        13           52          127
## 5        17           35          125
## 6        19           37          126
## 7        15           34          123
## 8        16           32          127
## 9        21           20          125
## 10       22           20          126
## 11       23           21          124
## 12       20           25          127
```

```
# Run the PCA using the 'princomp' function
DeerSpeciesAcousticFeaturesPCA <- princomp(DeerSpeciesAcousticFeatures[,-c(4)])
```

Then we will plot the results NOTE: each point represents one roar and the colors represent the class category.

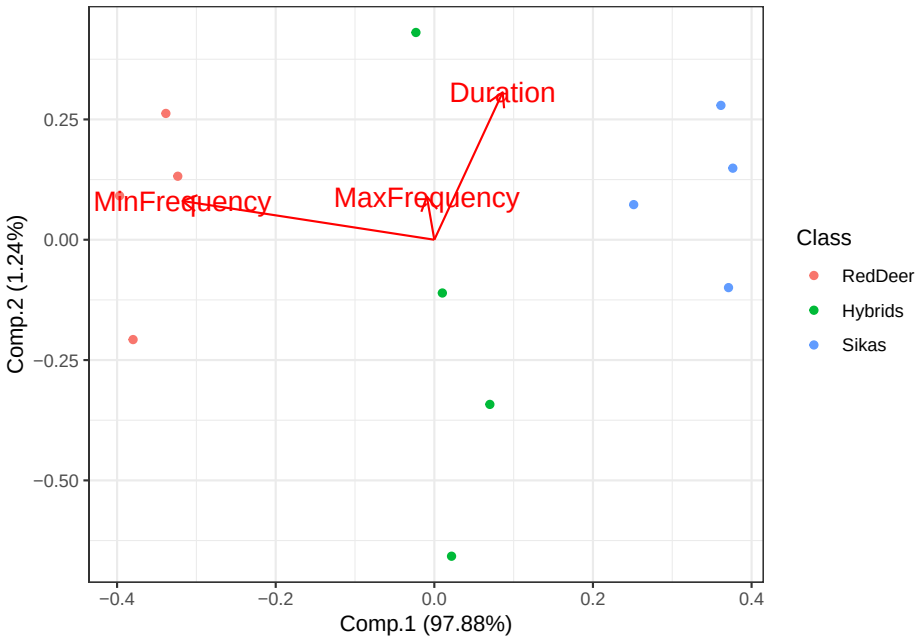
```
# Plot the results of our PCA.
ggplot2::autoplot(DeerSpeciesAcousticFeaturesPCA, data = DeerSpeciesAcousticFeatures, colour = 'C
```

```
         loadings = FALSE)+theme_bw()
```



What if we are interested in which features best distinguish between our groups? We can visulize this using the following code:

```
# Plot the PCA with arrows indicating which features are important for distinguishing
ggplot2::autoplot(DeerSpeciesAcousticFeaturesPCA, data = DeerSpeciesAcousticFeatures,
         loadings = TRUE, loadings.colour = 'red',
         loadings.label = TRUE,
         loadings.label.size = 5)+theme_bw()
```



This shows that the feature that best distinguishes between red deer and sika is the minimum frequency of their vocalizations. If you look on the x-axis (Comp.1) you can see that there is a substantial amount of separation between these groups.

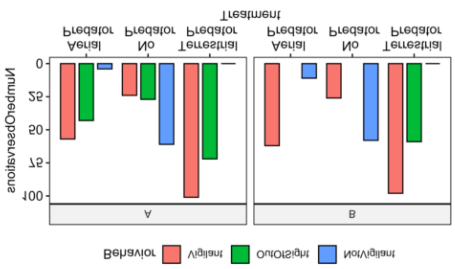
Please answer the following questions:

Question 6. In your own words, explain why we would want to use PCA for data visualization? What can we learn about the data when it is visualized this way?

Chapter 2

Lab 2. Activity Budgets and Ethograms

A



B

Background
In this lab you will continue to become familiar with the ways that we visualize and analyze behavioral data. For this lab I assume students have conducted Field Labs 1 and 2, so you should be familiar with how focal behavioral data are collected. Field Lab 1 data collection involves observation of animals in your own backyard, and the Field Lab 2 data collection uses methods outlined in: Hammond 2019, Vigilance behaviour in meerkats, ASAB Education.

Goals of the exercises
The main goal(s) of today's lab are to:

1) Enter and visualize your ethogram data
2) Calculate meerkat activity budgets 3) Compare inter-observer reliability from the meerkat data

Getting started
First we need to load the relevant packages for our data analysis. Packages contain all the functions that are needed for data analysis.

```
library(behaviouR)
```

Lab 2a. Enter and visualize your ethogram data

Here we will do an exploratory analysis of the ethograms that you created in Field Lab 1. First we will use some simulated (or made up) data to create our ethogram. In R we call the objects that contain our data 'dataframes'.

```
EthogramDF <-
  data.frame(Behavior=c('Resting','Locomotion','Foraging','Calling','Playing','Other')
                 TimesObserved=c(5,7,3,21,1,0))
```

If you run the object 'EthogramDF' you should see your table:

```
EthogramDF
```

```
##     Behavior TimesObserved
## 1    Resting             5
## 2 Locomotion             7
## 3   Foraging             3
## 4    Calling            21
## 5    Playing             1
## 6      Other             0
```

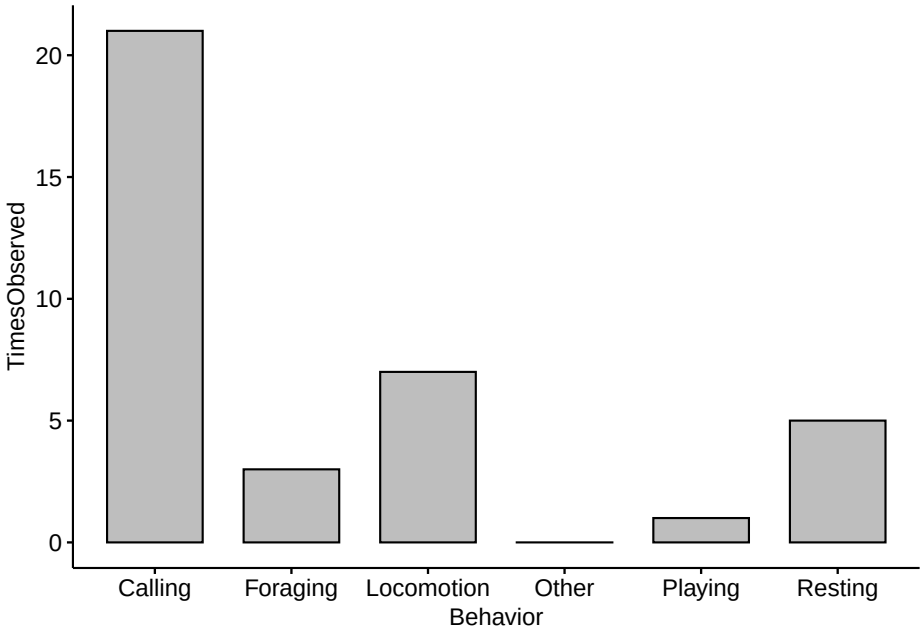
A common command to check data structure is 'str'. Here we can see our dataframe has two variables:'Behavior' and 'TimesObserved'.

```
str(EthogramDF)
```

```
## 'data.frame':    6 obs. of  2 variables:
##  $ Behavior     : Factor w/ 6 levels "Calling","Foraging",..: 6 3 2 1 5 4
##  $ TimesObserved: num  5 7 3 21 1 0
```

Now we can make a barplot to visualize our results using the 'ggpubr' package

```
ggbarplot(data=EthogramDF, x='Behavior', y='TimesObserved',fill='grey')
```



In the space below I want you to add in your own ethogram categories and create a plot based on the ethogram you created in Field lab 1.

Change the Behavior categories (i.e. Category1) to the categories you used in your ethogram and the TimesObserved values to the actual values you recorded

```
EthogramDFupdated <- data.frame(Behavior=c('Category1','Category2','Category3',
                                           'Category4','Category5','Category6'),
                                TimesObserved=c(1,2,3,4,5,6))
```

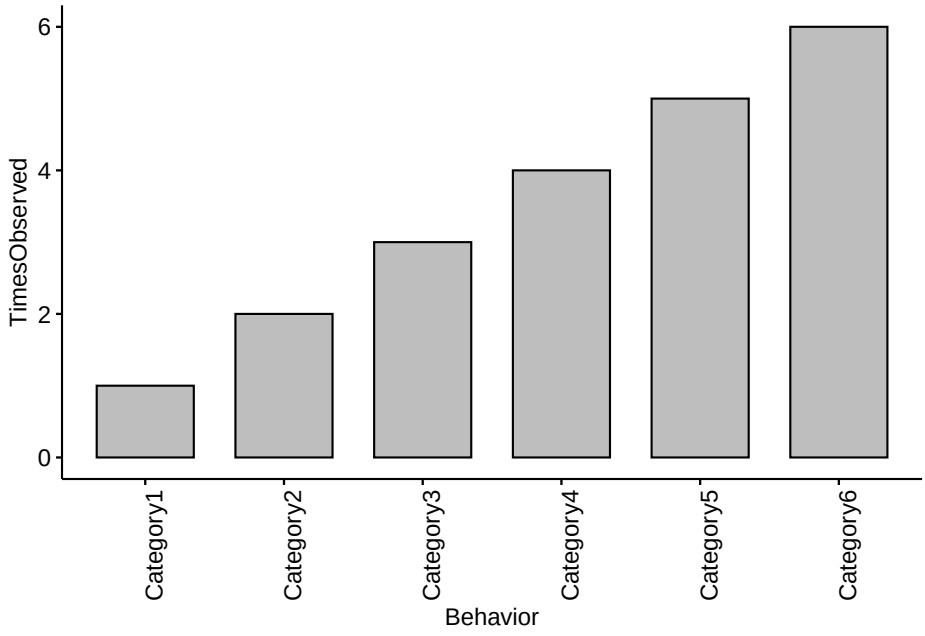
If you run the object you should see your dataframe

```
head(EthogramDFupdated)
```

```
##    Behavior TimesObserved
## 1 Category1             1
## 2 Category2             2
## 3 Category3             3
## 4 Category4             4
## 5 Category5             5
## 6 Category6             6
```

NOTE: Your plot will look different than this as you will enter your own data above!

```
ggbarplot(data=EthogramDFupdated, x='Behavior', y='TimesObserved',fill='grey',xtickslab.rt = 90)
```

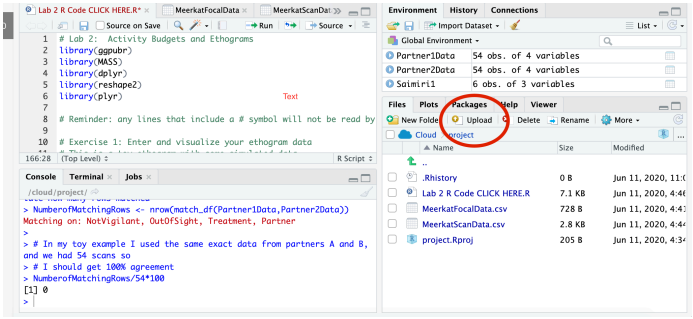


Question 1
Change the code to reflect the categories you used in your ethogram and change the TimesObserved values to the actual values you recorded. NOTE: You may need to add more categories if your ethogram had more than six. What trends did you notice in terms of behaviors observed?

Lab 2b. Calculate meerkat activity budgets.

Our previous ethograms only included counts of different behaviors that we observed, but we did not standardize our results in any way. Activity budgets give an indication of how much time an animal spends doing each activity; these are generally expressed as percentages.

First we need to import the data. If you are using RStudio Cloud you can import your data by clicking the 'upload' button and find the datasheet saved on your computer.



We also have sample data saved in the package so we can load that using the following code

NOTE: You will want to keep the file name the exact same or R won't be able to find the data. Let's check the structure of our data.

```
str(MeerkatFocalData)
```

```
## 'data.frame':    28 obs. of  6 variables:
##  $ StartTimeMin            : int  0 1 2 3 3 4 4 5 5 5 ...
##  $ StartTimeSec            : int  30 20 40 0 30 0 20 0 30 50 ...
##  $ BehaviorCode            : Factor w/ 5 levels "Drinking","Eating",..: 2 4 1 1 2 3 2 3 3 3 ..
##  $ TimeSeconds             : int  30 80 160 180 210 240 260 300 330 350 ...
##  $ SecondsEngagedinBehavior: int  30 50 80 20 30 30 20 40 30 20 ...
##  $ Partner                 : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
```

Then we calcuate the total number of seconds for each behavior. There are many different ways that we can summarize our data using R, but we will use the package 'dplyr'.

Here we take the sum for each unique behavior and return it

```
MeerkatFocalData %>%
  dplyr::group_by(BehaviorCode) %>%
  dplyr::summarise(TotalSeconds = sum(SecondsEngagedinBehavior))
```

```
## # A tibble: 5 x 2
##   BehaviorCode TotalSeconds
##   <fct>               <int>
## 1 Drinking              200
## 2 Eating                150
## 3 Foraging              350
## 4 Locomotion            250
## 5 Sentry                240
```

Then we need to save the output as an R object

```
MeerkatFocalDataSummary <- MeerkatFocalData %>%
  dplyr::group_by(BehaviorCode) %>%
```

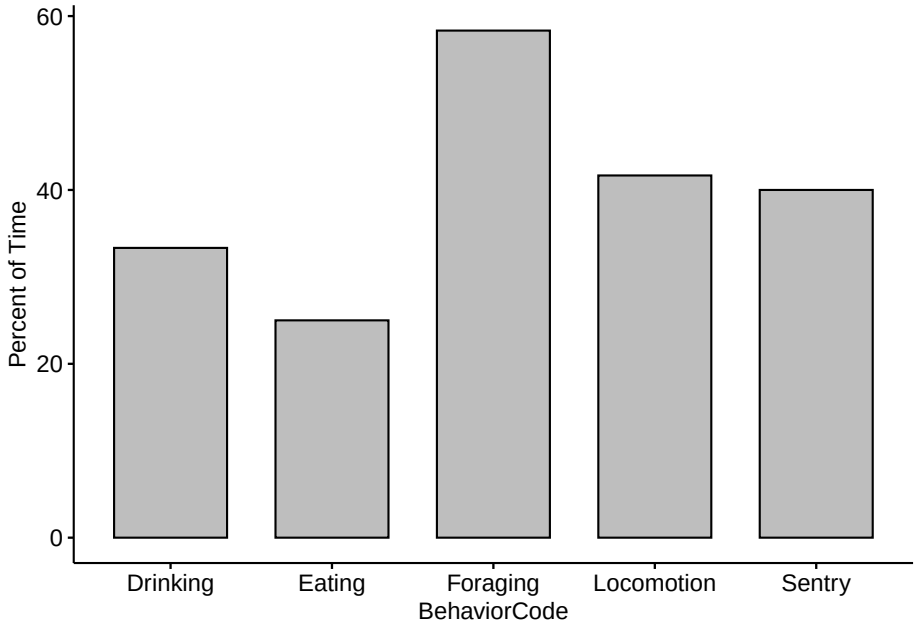
```
  dplyr::summarise(TotalSeconds = sum(SecondsEngagedinBehavior))
```

Now we need to standardize for our total observation time. We divide by 600 because our video was 10 minutes (or 600 seconds) long.

```
MeerkatFocalDataSummary$ActivityBudget <- MeerkatFocalDataSummary$TotalSeconds/600*100
```

Now let's plot our results! NOTE: The example here is using fake data.

```
ggbarplot(data=MeerkatFocalDataSummary,x='BehaviorCode',y='ActivityBudget',fill='grey'
```



Now lets see if there were differences between you and your partner. We need to save the output as an R object, but this time including the sums by partner.

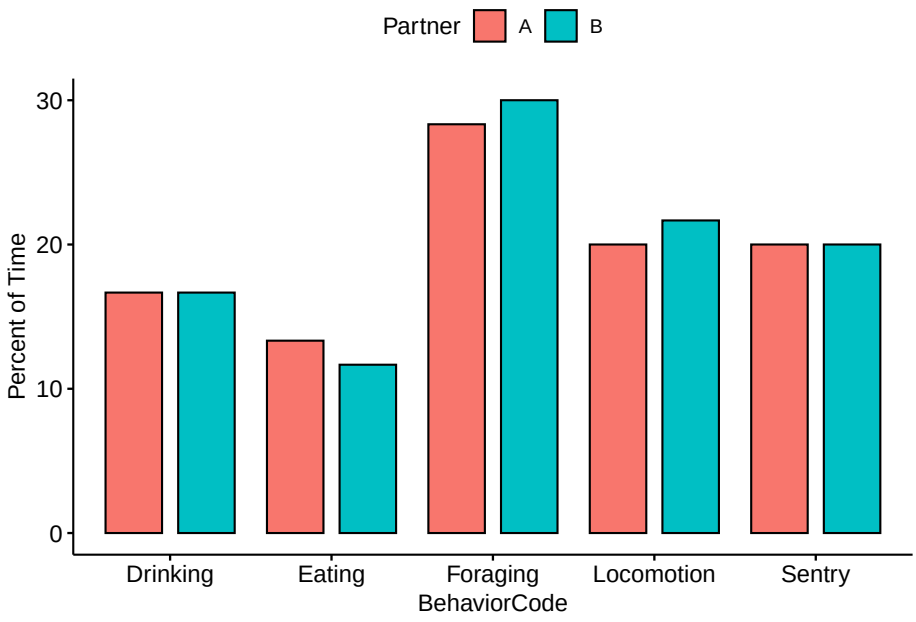
```
MeerkatFocalDataPartner <- MeerkatFocalData %>%
  dplyr::group_by(BehaviorCode,Partner) %>%
  dplyr::summarise(TotalSeconds = sum(SecondsEngagedinBehavior))
```

Again, we divide by 600 because our video was 10 minutes (or 600 seconds) long. We then multiply by 100 so that we can report in percentages.

```
MeerkatFocalDataPartner$ActivityBudget <- MeerkatFocalDataPartner$TotalSeconds/600*100
```

Now we compare our data with our partner's. Note that there are two new lines in the code below. The fill argument tells R which category or factor to use to color the bars. The position argument tells R to place the bars side-by-side.

```
ggbarplot(data=MeerkatFocalDataPartner,x='BehaviorCode',y='ActivityBudget', fill='Partn
          position = position_dodge(0.9))+ylab('Percent of Time')
```



Question 3.
Were there any noticeable differences between you and your partner's activity budgets?

Lab 2c. Scan sampling and inter-observer reliability.

Load your data as you did before, but this time upload the 'MeerkatScanData.csv' file.

Check the structure of our data. If we use 'head' it will return the first few rows of our dataframe.

```
head(MeerkatScanData)
```

```
##   Time Vigilant NotVigilant OutOfSight  Treatment Partner
## 1   10        1           3          2 NoPredator       A
## 2   20        1           5          0 NoPredator       A
## 3   30        2           3          1 NoPredator       A
## 4   40        2           4          0 NoPredator       A
## 5   50        1           4          1 NoPredator       A
## 6   60        2           4          1 NoPredator       A
```

For this analysis the 'Time' column isn't needed so we can remove it; It is the

first column so we use 1 in the code below.

```
MeerkatScanData <-MeerkatScanData[,-c(1)]
```

R deals with blanks in dataframes by converting them to 'NA'. This is useful for some cases, but we want to convert our blanks to zeros using the following code.

```
MeerkatScanData[is.na(MeerkatScanData)] <- 0
```

Here we take the sum for each unique behavior and return it.

```
MeerkatScanData %>%
  dplyr::group_by(Treatment) %>%
  dplyr::summarise(Vigilant = sum(Vigilant),
                   OutOfSight=sum(OutOfSight),
                   NotVigilant=sum(NotVigilant))
```

```
## # A tibble: 3 x 4
##   Treatment           Vigilant OutOfSight NotVigilant
##   <fct>                  <int>      <dbl>       <int>
## 1 AerialPredator           119         73          15
## 2 NoPredator                50         64         119
## 3 TerrestrialPredator      199        131           0
```

We need to save the output as an R object

```
MeerkatScanDataSummary <- MeerkatScanData %>%
  dplyr::group_by(Treatment,Partner) %>%
  dplyr::summarise(Vigilant = sum(Vigilant),
                   OutOfSight=sum(OutOfSight),
                   NotVigilant=sum(NotVigilant))
```

There are many functions in R that allow us to change the format of our data. This one changes the format to one that is more easily used by 'ggpubr'

```
MeerkatScanDataSummaryLong <- reshape2::melt(MeerkatScanDataSummary, id.vars=c("Treatme
str(MeerkatScanDataSummaryLong)
```

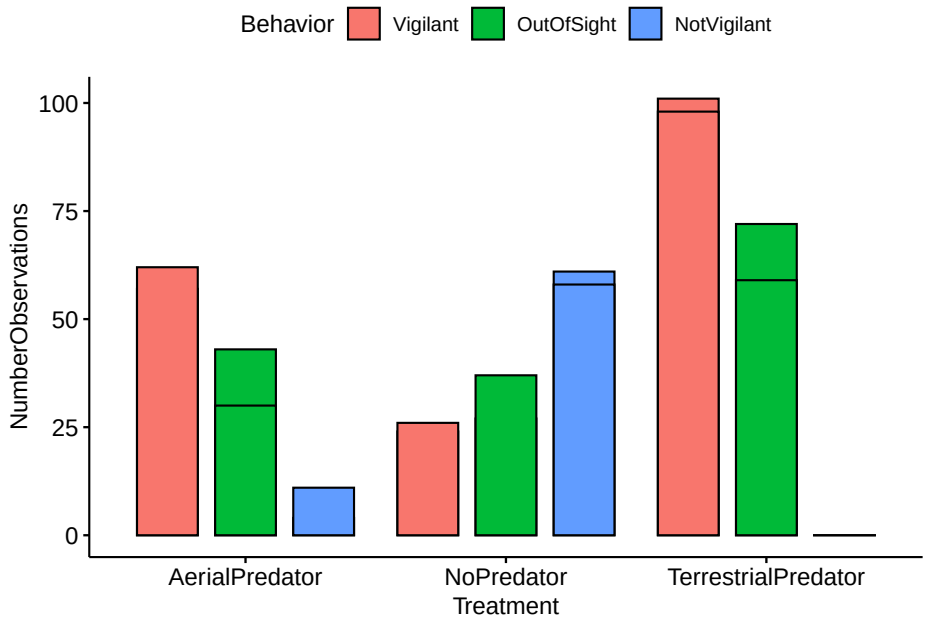
```
## 'data.frame':    18 obs. of  4 variables:
##  $ Treatment: Factor w/ 3 levels "AerialPredator",..: 1 1 2 2 3 3 1 1 2 2 ...
##  $ Partner  : Factor w/ 2 levels "A","B": 1 2 1 2 1 2 1 2 1 2 ...
##  $ variable : Factor w/ 3 levels "Vigilant","OutOfSight",..: 1 1 1 1 1 1 2 2 2 2 ..
##  $ value    : num  57 62 24 26 101 98 43 30 27 37 ...
```

Now we change the column names so they are better for plotting

```
colnames(MeerkatScanDataSummaryLong) <- c('Treatment','Partner','Behavior','NumberObse
```

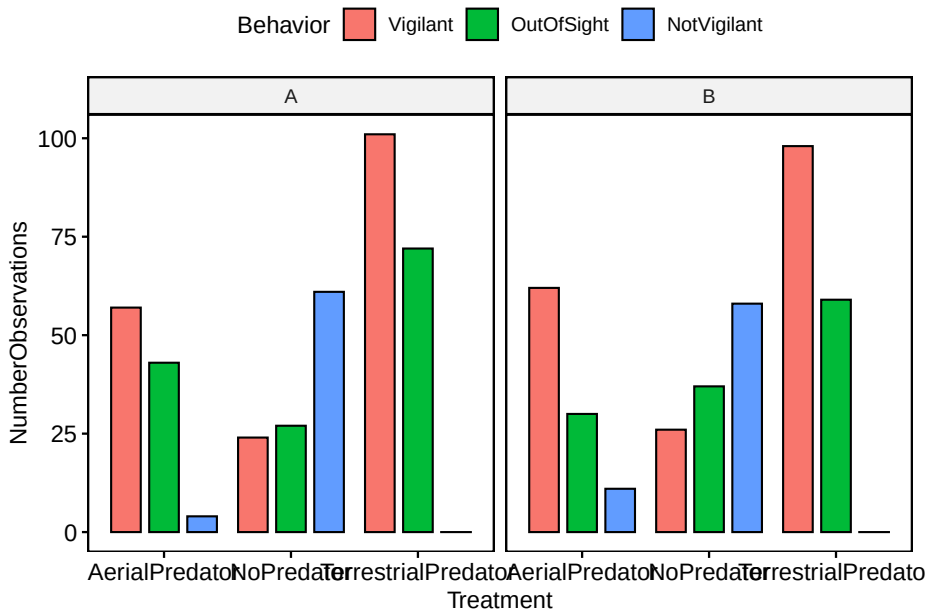
Again we can plot our results

```
ggbarplot(data=MeerkatScanDataSummaryLong,x='Treatment',y='NumberObservations', fill='Behavior',
          position = position_dodge(0.9))
```



And now we can visually compare our results with our partners. Note that in order to separate the plots by partner we use facet.by = 'Partner'.

```
ggbarplot(data=MeerkatScanDataSummaryLong,x='Treatment',y='NumberObservations', fill='Behavior',
          position = position_dodge(0.9),facet.by = 'Partner')
```



Now we want to calculate inter-observer reliablity. NOTE: that you will want to change the A and B to the names you used in the datasheet

```
Partner1Data <- subset(MeerkatScanData,Partner=='A')
Partner1Data <- Partner1Data[,-c(5)] # We remove the 'Partner' column so that we only

Partner2Data <- subset(MeerkatScanData,Partner=='B')
Partner2Data <- Partner2Data[,-c(5)] # We remove the 'Partner' column so that we only
```

Reliability is often and most simply expressed as a correlation coefficient. A correlation of 1 means a perfect positive association between two sets of measurements, whereas a correlation of zero means the complete absence of a linear association. Reliability is generally calcuated for each category of behavior.

```
# Here we calculate reliability for vigilance
VigilantCorrelation <- cor(Partner1Data$Vigilant,Partner2Data$Vigilant)

# Here we calculate reliability for vigilance
NotVigilantCorrelation <- cor(Partner1Data$NotVigilant,Partner2Data$NotVigilant)

# Here we calculate reliability for out of sight
OutOfSightCorrelation <-  cor(Partner1Data$OutOfSight,Partner2Data$OutOfSight)

# Print the results
cbind.data.frame(VigilantCorrelation,NotVigilantCorrelation,OutOfSightCorrelation)
```

```
##   VigilantCorrelation NotVigilantCorrelation OutOfSightCorrelation
```

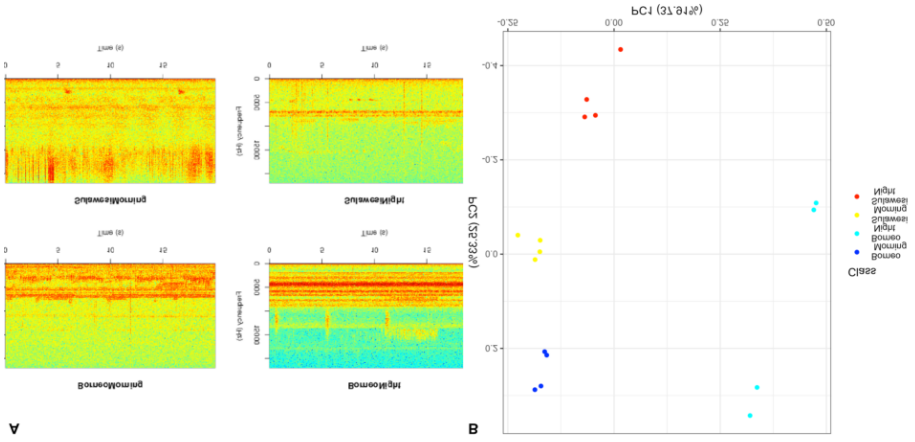
```
## 1          0.9027009          0.9347273          0.7840471
```

NOTE: The correlation values between you and your partner will be different!

Question 4.
What was the reliability (or correlation coefficent) between you and your partner for each of the different behavioral categories? What do you think lead to these differences?

Chapter 3

Lab 3. Analyzing Acoustic Data



Background
In this lab you will become familiar with the ways that we analyze acoustic data. You will start by analyzing focal recordings and soundscapes from Southeast Asia, and you will then upload the data you collected for Field Lab 3 and do some prelminary analysis.

Goals of the exercises
The main goal(s) of today's lab are to:
1) Create your own spectrograms
2) Learn how we use PCA to visualize differences in acoustic data

3) Upload and visualize your own data

Getting started
First we need to load the relevant packages for our data analysis. Packages contain all the functions that are needed for data analysis. First we load the required libraries

```
library(behaviouR)
```

We need to do some data prep to work through the examples

```
# First you can check where your working directory is; this is where R will store the
getwd()

# Now we will create a file at this location called 'FocalRecordings'
dir.create(file.path('FocalRecordings'), showWarnings = FALSE)

# Now we will load the sound files that were in the R package
data("FocalRecordings")

# Let's check the structure
head(FocalRecordings)

# Now we will save the recordings to the new folder you created
for(a in 1:length(FocalRecordings)){
  FileName <- FocalRecordings[[a]][1][[1]]
  WaveFile <- FocalRecordings[[a]][2][[1]]
  tuneR::writeWave(WaveFile, paste("FocalRecordings/",FileName,sep=''))
}
```

Part 1. Loading a sound file and making a spectrogram

First we want to check which files are in the folder There are five gibbon female recordings and five great argus recordings.

```
list.files("FocalRecordings")
```

```
##  [1] "FemaleGibbon_1.wav" "FemaleGibbon_2.wav" "FemaleGibbon_3.wav"
##  [4] "FemaleGibbon_4.wav" "GreatArgus_1.wav"   "GreatArgus_2.wav"
##  [7] "GreatArgus_3.wav"   "GreatArgus_4.wav"   "MaleSolo_1.wav"
## [10] "MaleSolo_2.wav"     "MaleSolo_3.wav"     "MaleSolo_4.wav"
## [13] "Thumbnails"
```

Now let's read in one of the files for further investigation. We do that with the function 'readWave'. The .wav stands for 'Waveform Audio File' and is a

common format that scientists use to save their sound files.

```
GibbonWaveFile <- tuneR::readWave("FocalRecordings/FemaleGibbon_1.wav")
```

Now if we run the object it will return some information

```
GibbonWaveFile
```

```
##
## Wave Object
##  Number of Samples:      572054
##  Duration (seconds):     12.97
##  Samplingrate (Hertz):   44100
##  Channels (Mono/Stereo): Mono
##  PCM (integer format):   TRUE
##  Bit (8/16/24/32/64):    16
```

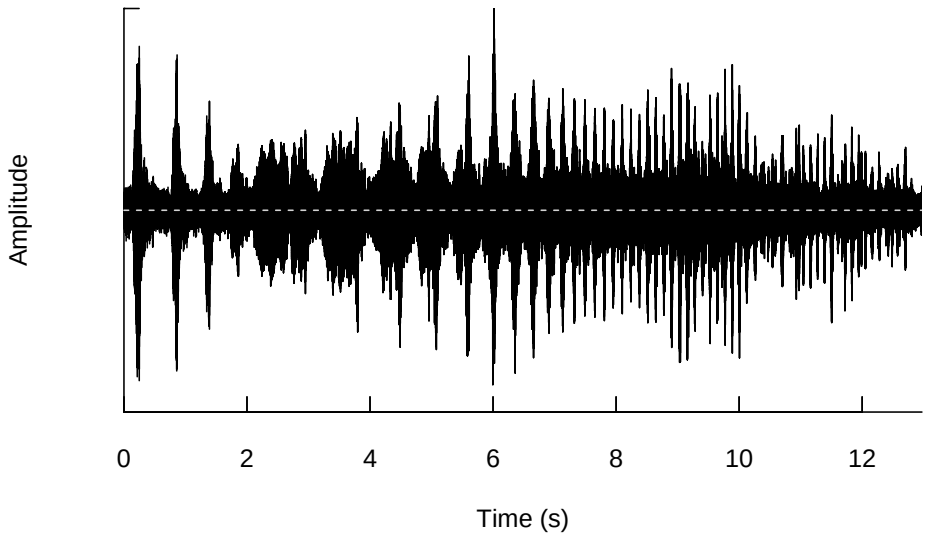
What we see is that the soundfile is ~13 seconds long, was recorded at a sampling rate of 44100 samples per second and has a total of 572054 samples. We can check if that makes sense using the following equation (duration* sampling rate).

```
seewave::duration(GibbonWaveFile)* GibbonWaveFile@samp.rate
```

```
## [1] 572054
```

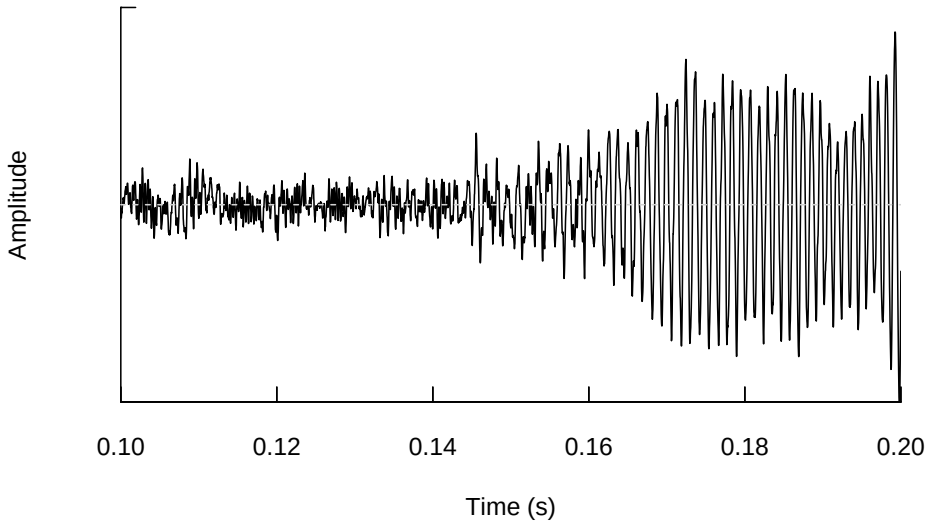
Now we can plot the waveform from our sound file using the following code:

```
seewave::oscillo(GibbonWaveFile)
```



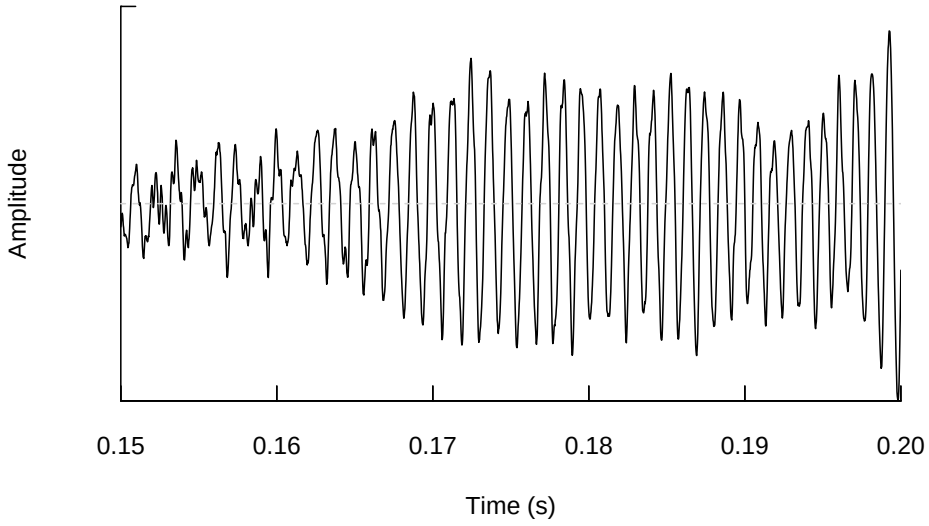
We can also zoom in so that we can actually see the shape of the wave.

```
seewave::oscillo(GibbonWaveFile, from=0.1, to=0.2)
```



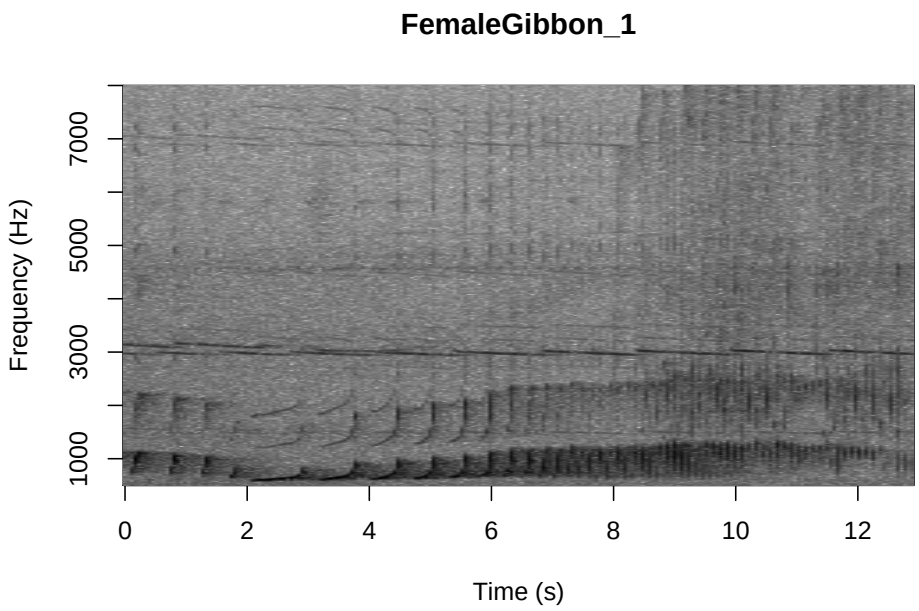
And if we zoom in again we see the wave shape even more.

```
seewave::oscillo(GibbonWaveFile, from=0.15, to=0.2)
```



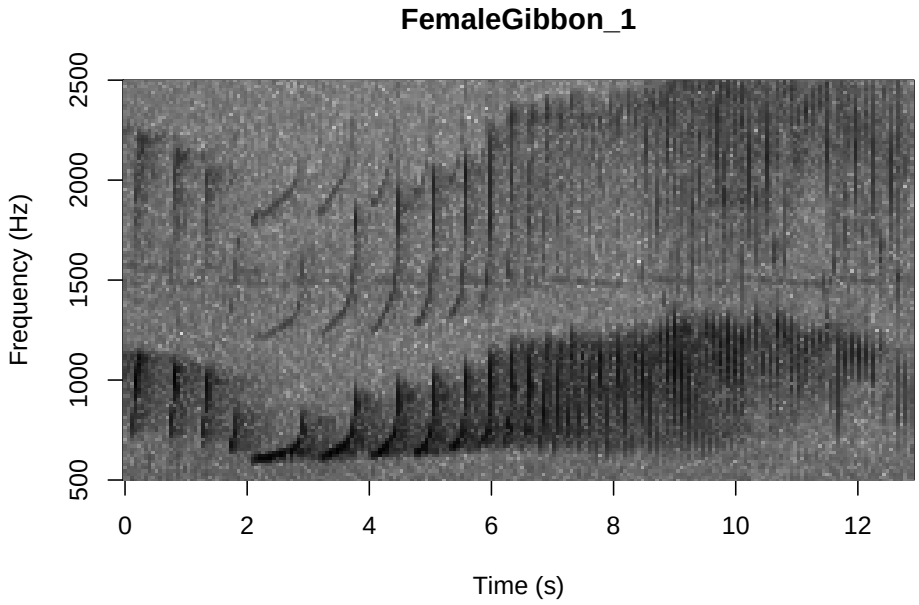
As we learned in class this week, a common way for scientists to study animal sounds is through the use of spectrograms. Here we will plot a spectrogram of a single gibbon call.

```
SpectrogramSingle(sound.file ="FocalRecordings/FemaleGibbon_1.wav")
```



There are many different things you can change when creating a spectrogram. In the above version of the plot there is a lot of space above the gibbon calls, so we will change the frequency range to better visualize the gibbons.

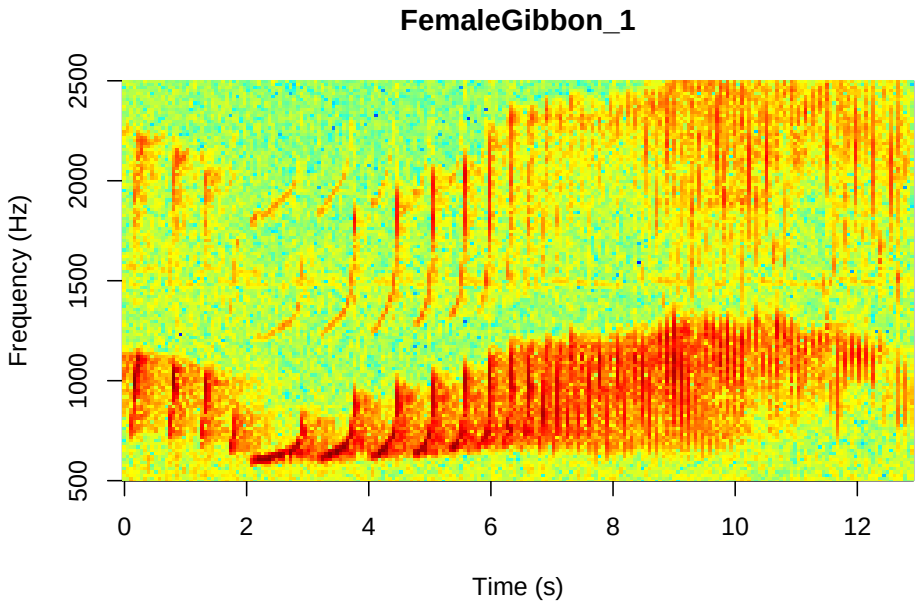
```
SpectrogramSingle(sound.file ="FocalRecordings/FemaleGibbon_1.wav",
                  min.freq = 500,max.freq=2500)
```



In the code we are using there is also the option to change the color of the

spectrogram adding the following code:

```
SpectrogramSingle(sound.file ="FocalRecordings/FemaleGibbon_1.wav",
                  min.freq = 500,max.freq=2500,
                  Colors = 'Colors')
```



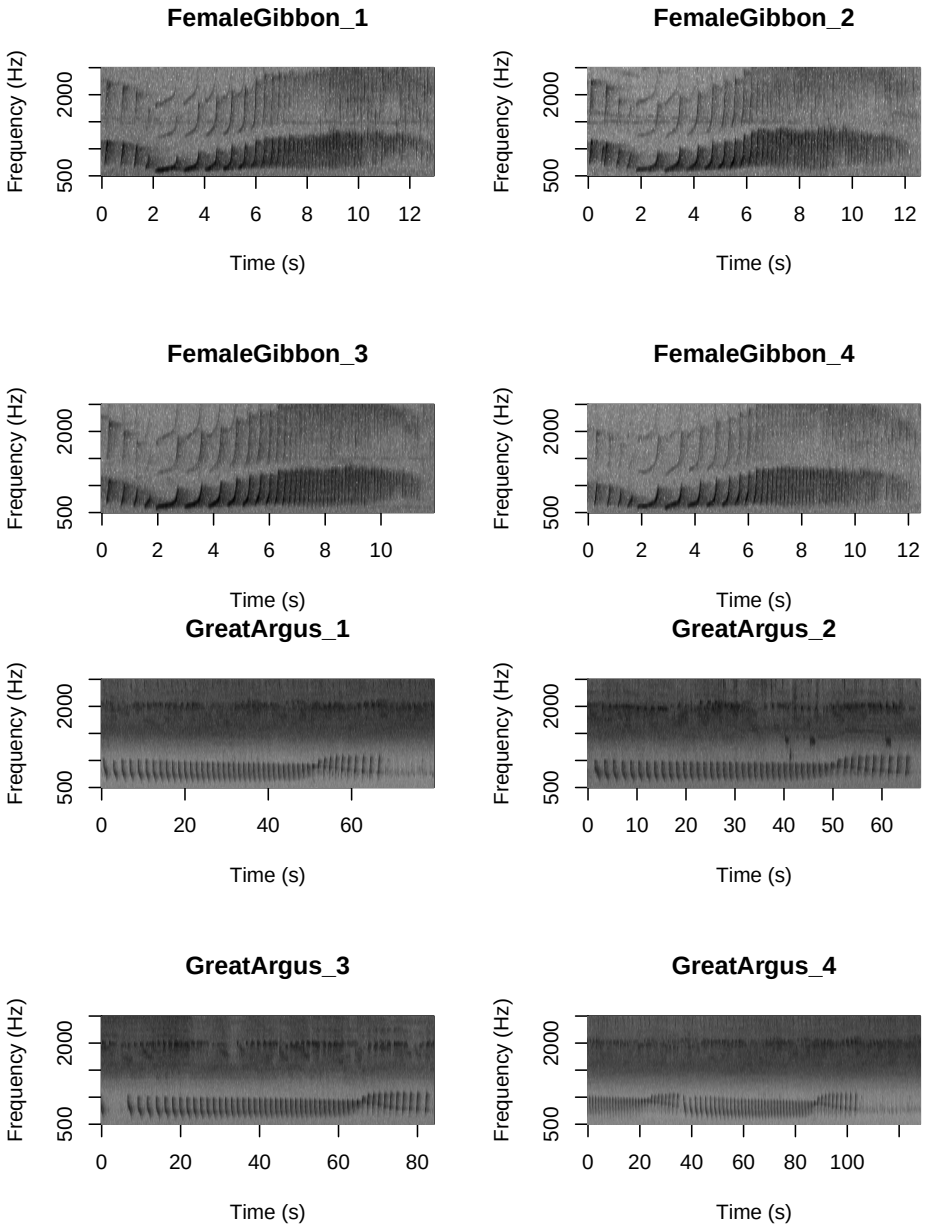
NOTE: Changing the colors doesn't change how we read the spectrograms,and different people have different preferences.

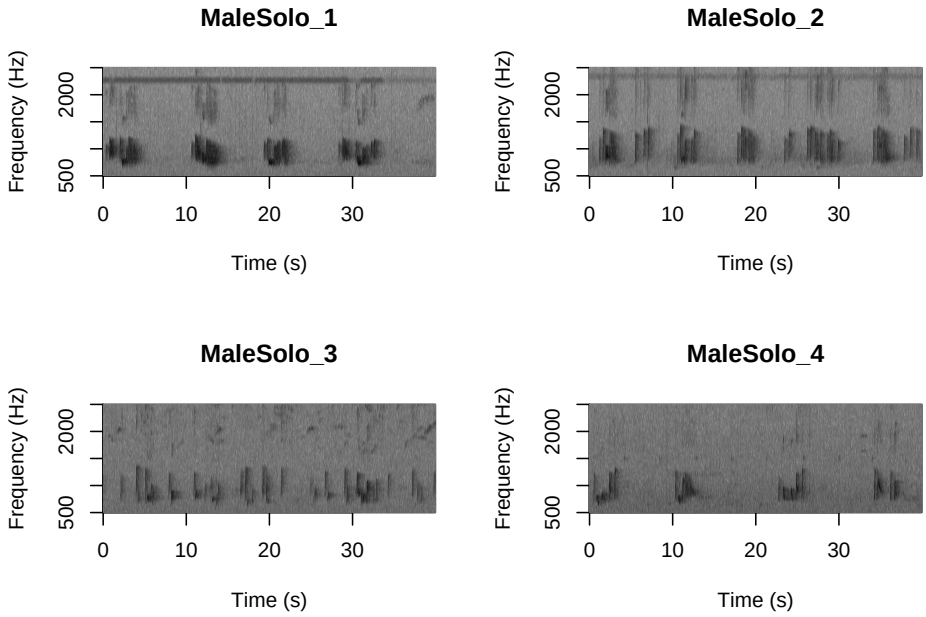
Now we can plot all of the spectrograms in our FocalRecordings file at once.

NOTE: Use the navigation buttons on the plots tab to move back and forth between the different spectrograms.

```
# We can tell R to print the spectrograms 2x2 using the code below
par(mfrow=c(2,2))

# This is the function to create the spectrograms
SpectrogramFunction(input.dir = "FocalRecordings",min.freq = 500,max.freq=2500)
```


FemaleGibbon_1
FemaleGibbon_2
FemaleGibbon_3
FemaleGibbon_4
GreatArgus_1
GreatArgus_2
GreatArgus_3
GreatArgus_4
Frequency (Hz)
Time (s)



Question 1. What differences do you notice about gibbon and great argus spectrograms?

Now that you are doing looking at the spectrograms we will clear the space. You can always re-run the code to look at the again.

```
graphics.off()
```

Part 2. Visualizing differences in gibbon and great argus calls

A major question in many different types of behavioral studies is whether there are differences between groups. This is also the case for acoustic data, where researchers may be interested if there are differences between sexes, populations, individuals, or vocalizations emitted in different contexts. Here we will work through a simple example where we will investigate differences in gibbon and argus calls.

The first step that we need to do is called feature extraction. There are many different ways that scientists do this, but the overarching idea is that sound data contains too much redundant information to be used on a model. Computers just don't have enough power to crunch all those numbers, so we need to identify a meaningful set of features that is smaller than using the while waveform. This is also the case for image processing.

We are going to use a feature extraction method called 'Mel-frequency cepstral coefficients'. I will not expect you to know any more about them, apart from the fact they are very useful for human speech and bioaccoustics applications.

Below is the function to extract features from our focal recordings

```
FeatureDataframe <- MFCCFunction(input.dir = "FocalRecordings")
```

In this new dataframe each row represents one gibbon or great argus call. Let's check the dimensions of this new dataframe.

```
dim(FeatureDataframe)
```

```
## [1]  12 178
```

This shows us that for each of the 10 calls there are 178 features. This is in contrast to a sound file. Let's check again to remind ourselves.

```
GibbonWaveFile
```

```
##
## Wave Object
##  Number of Samples:      572054
##  Duration (seconds):     12.97
##  Samplingrate (Hertz):   44100
##  Channels (Mono/Stereo): Mono
##  PCM (integer format):   TRUE
##  Bit (8/16/24/32/64):    16
```

This sound file is comprised of 572054 numbers. Now let's check the first row of our new dataframe which contains the new features for the same gibbon call.

```
# This isolates the first row.
FeatureDataframe[1,]
```

```
##          Class         1         2         3        4         5         6
## 1 FemaleGibbon -4.914312 -16.40678 -4.672827 20.59783 -4.632383 -6.999979
##          7         8        9       10        11        12        13        14
## 1 4.438287 -3.773641 -5.61846 10.73634 0.9757686 -1.163231 -6.800348 -17.70036
##          15       16       17       18       19        20        21        22
## 1 -2.564371 5.702963 8.604349 5.529339 2.561141 -2.507278 -8.423425 -6.229291
##          23        24        25        26      27       28       29       30
## 1 -3.115276 -7.588447 -23.27116 -3.826297 17.3082 8.632006 -1.60833 2.771364
##          31        32        33        34        35        36       37       38
## 1 -3.018685 -13.47489 -5.147027 -6.892077 -12.00958 -21.92954 9.776812 14.48623
##           39       40       41        42      43      44        45        46
## 1 0.08553639 2.275835 4.157477 -15.22412 -8.4252 8.55771 -7.545372 -20.37116
##          47       48        49       50       51        52        53       54
## 1 -9.496601 20.42913 0.1681313 2.009451 4.554415 -16.20235 -2.036649 9.715715
##          55        56       57        58       59        60       61       62
```

```
## 1 -1.521847 -4.390747 -23.6537 -1.739798 13.12778 -5.910114 6.569014 6.026896
##          63       64       65        66        67        68       69       70
## 1 -9.226613 -1.68419 4.550749 -5.170068 -10.94638 -20.08074 2.986301 9.424681
##          71       72      73        74         75       76       77        78
## 1 -6.254541 10.38071 -5.0744 -4.867023 -0.4075844 -2.65999 2.784278 -11.88797
##          79       80       81        82       83        84      85       86
## 1 -13.21431 5.279703 8.139582 -5.553872 5.027628 -3.739047 -3.1116 4.341732
##          87      88        89       90        91        92       93       94
## 1 -1.209793 1.00652 -1218.033 -1112.45 -830.5594 -462.4005 25.70725 66.06332
##          95        96       97       98       99       100      101      102
## 1 -5.735883 -36.24788 53.94479 37.60848 3.826036 -1222.493 -1046.01 -770.889
##         103      104      105     106       107       108       109       110
## 1 -393.6294 80.33639 49.16433 5.84342 -92.54861 -126.7318 -115.9133 -78.65571
##         111       112       113       114      115      116      117       118
## 1 -1222.167 -1044.567 -793.9816 -401.6897 87.24976 31.49047 -2.96363 -118.2727
##         119       120       121       122       123       124       125
## 1 -154.4816 -83.69734 -33.83878 -1229.136 -1086.172 -808.4772 -399.4776
##        126      127      128       129     130      131      132       133
## 1 53.89249 11.00474 11.99423 -57.18338 1.68353 85.31662 102.8748 -1239.725
##        134       135       136      137      138       139       140       141
## 1 -1088.64 -803.1513 -479.0124 44.22369 73.85069 -22.86258 -84.05236 -3.588271
##         142      143       144       145       146       147      148      149
## 1 -2.478177 9.528495 -1240.362 -1068.622 -782.7229 -440.9671 63.08211 80.21284
##         150       151       152    153       154       155       156       157
## 1 -46.79594 -82.71886 -40.87362 -75.92 -49.31079 -1285.428 -1091.577 -842.4694
##         158      159     160       161       162      163      164     165
## 1 -460.0004 72.63099 32.0981 -40.61593 -20.85975 31.29035 11.68475 66.2166
##         166       167       168       169     170      171       172       173
## 1 -1225.674 -1065.805 -823.5072 -452.4384 74.0775 24.51663 -33.48887 -13.24513
##        174      175      176      177
## 1 25.57118 3.053292 26.88251 12.97175
```

```
# This tells us how many features we have.
ncol(FeatureDataframe[1,])
```

```
## [1] 178
```

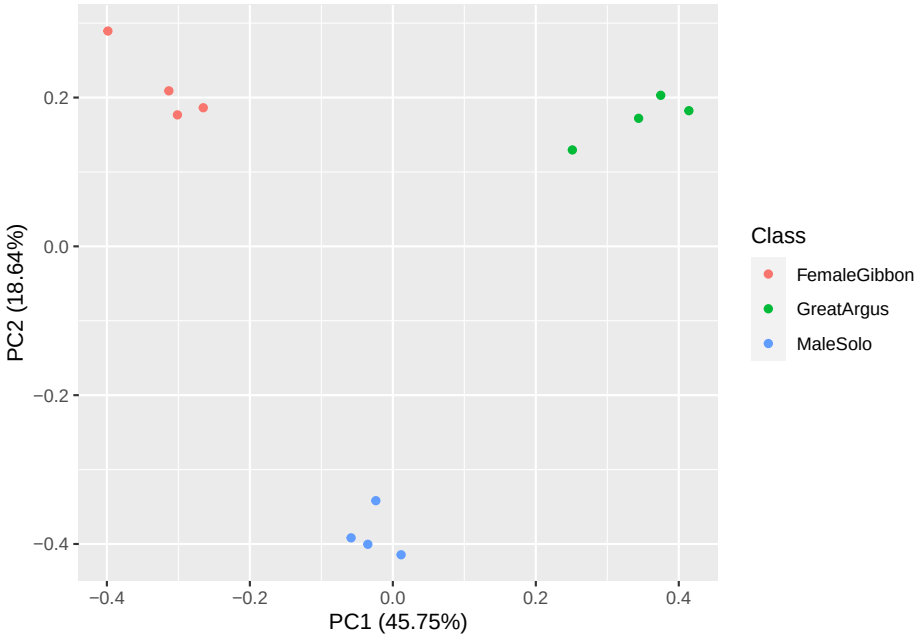
OK, now we are ready to do some plotting. If you remember from our earlier lab, the data structure we have here is multivariate, as each call has multiple values associated with it. So we will use the same approach (principal component analysis) to visualize our gibbon and great argus calls.

First we run the principal component analysis

```
pca_res <- prcomp(FeatureDataframe[,-c(1)], scale. = TRUE)
```

Now we visualize our results

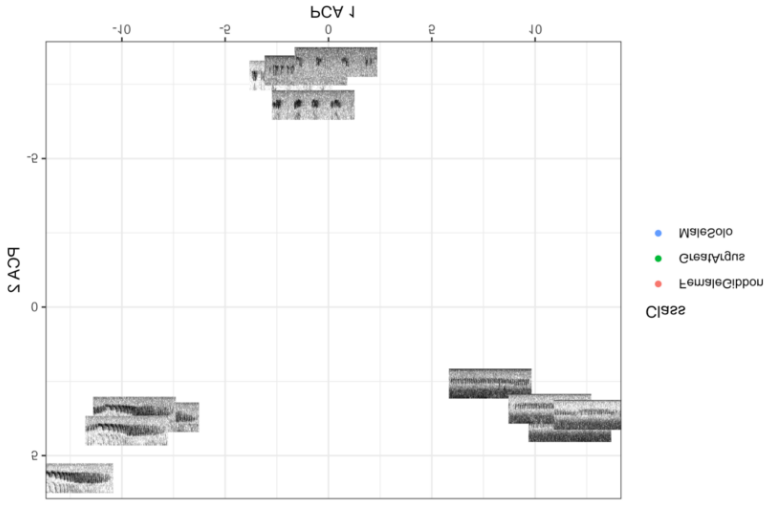
```
ggplot2::autoplot(pca_res, data = FeatureDataframe,
         colour = 'Class')
```



What we see in this plot is strong clustering, with the gibbon calls closer to each other than the great argus calls. There is one great argus call that seems to be a bit of an outlier. Let's check that out using the code below.

NOTE: The code below may take a few minutes to run. If you can't see the plot well click the 'zoom' button on the plot tab

```
BiplotAddSpectrograms(input.dir.Focal="FocalRecordings")
```



Now that you are done looking at the figure we will clear the space. You can always re-run the code to look at the again.

```
graphics.off()
```

Question 2. Do you see any differences in the spectrogram of the great argus calls that are clustered together and the outlier?

Part 3. Soundscapes

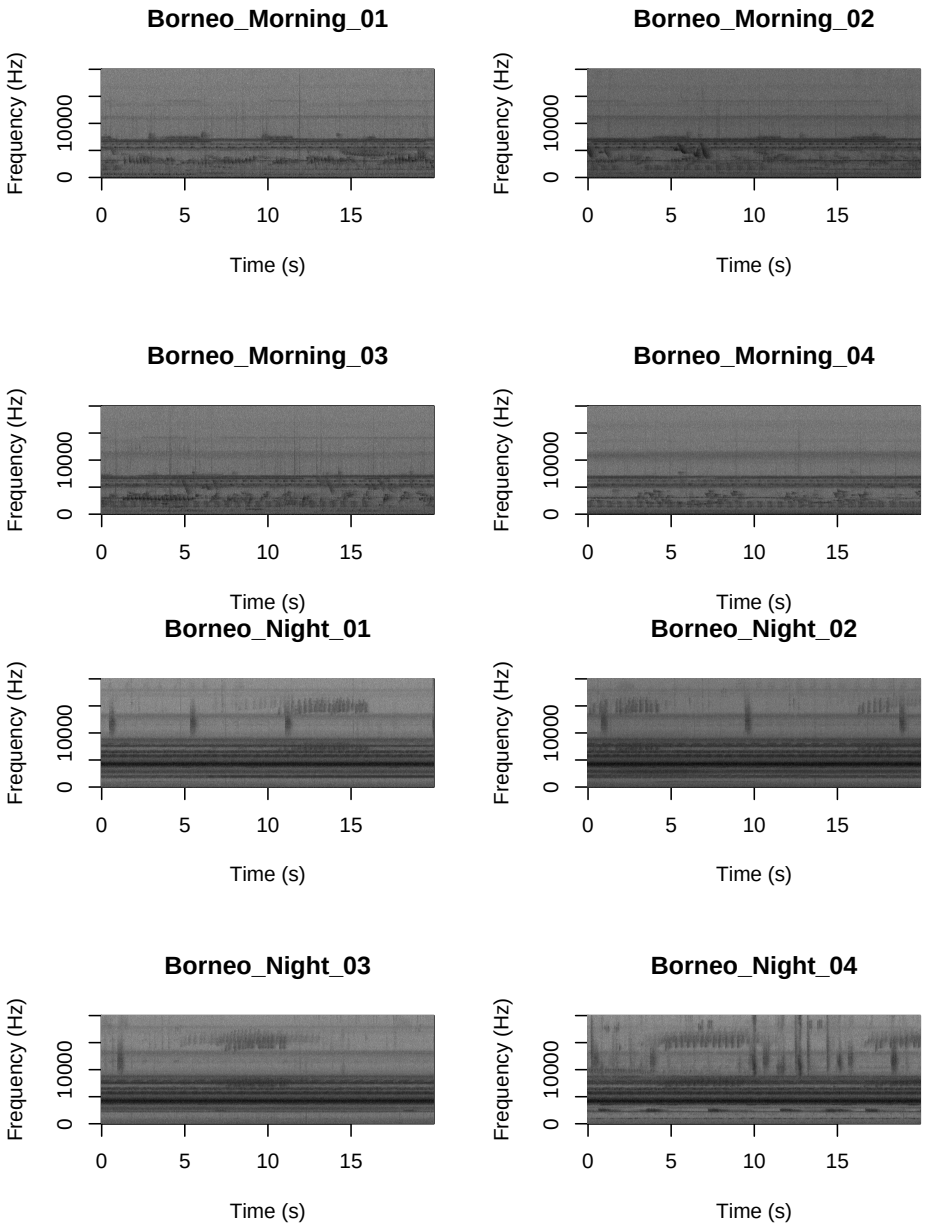
Now we will do something similar to our investigation of the focal recordings, but using a soundscape example. Here we have 20-sec recordings taken from two different locations (Borneo and Sulawesi) and two different times (06:00 and 18:00). Let's check what is in the folder.

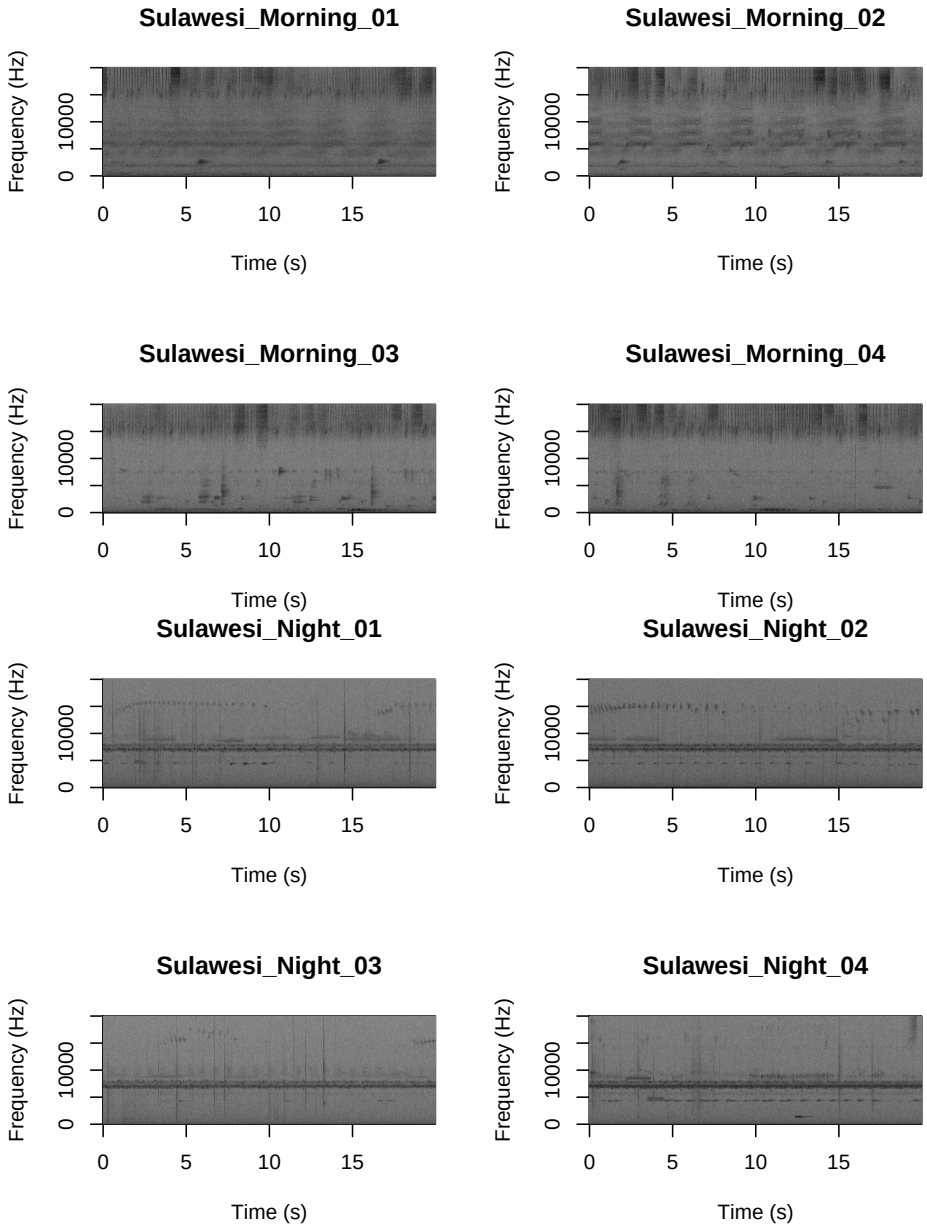
```
list.files("SoundscapeRecordings")
```

```
##  [1] "Borneo_Morning_01.wav"   "Borneo_Morning_02.wav"
##  [3] "Borneo_Morning_03.wav"   "Borneo_Morning_04.wav"
##  [5] "Borneo_Night_01.wav"     "Borneo_Night_02.wav"
##  [7] "Borneo_Night_03.wav"     "Borneo_Night_04.wav"
##  [9] "Sulawesi_Morning_01.wav" "Sulawesi_Morning_02.wav"
## [11] "Sulawesi_Morning_03.wav" "Sulawesi_Morning_04.wav"
## [13] "Sulawesi_Night_01.wav"   "Sulawesi_Night_02.wav"
## [15] "Sulawesi_Night_03.wav"   "Sulawesi_Night_04.wav"
```

Now we will create spectrograms for each recording

```
par(mfrow=c(2,2))
SpectrogramFunctionSite(input.dir = "SoundscapeRecordings",
                        min.freq = 0,max.freq=20000)
```





Now that you are doing looking at the figure we will clear the space. You can always re-run the code to look at the again.

```
graphics.off()
```

Now just as above we will do feature extraction of our soundscape recordings. Thiswill convert our dataset into a smaller, much more manageable size.

```
SoundscapeFeatureDataframe <-
  MFCCFunctionSite(input.dir = "SoundscapeRecordings")
```

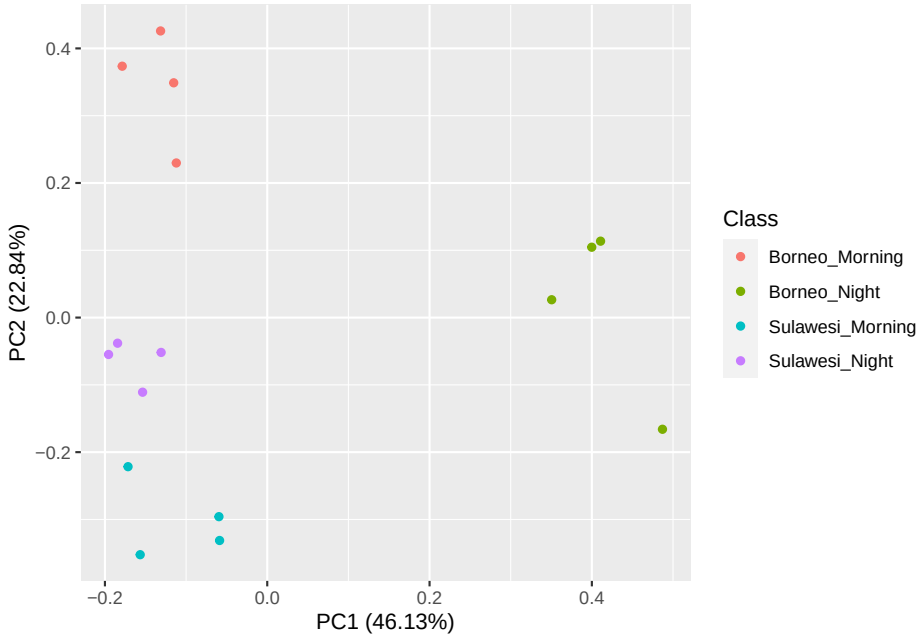
Check the resulting structure of the dataframe to make sure it looks OK.

```
dim(SoundscapeFeatureDataframe)
```

```
## [1]  16 177
```

Now we visualize our results

```
pca_res <- prcomp(SoundscapeFeatureDataframe[,-c(1)], scale. = TRUE)
ggplot2::autoplot(pca_res, data = SoundscapeFeatureDataframe,
         colour = 'Class')
```



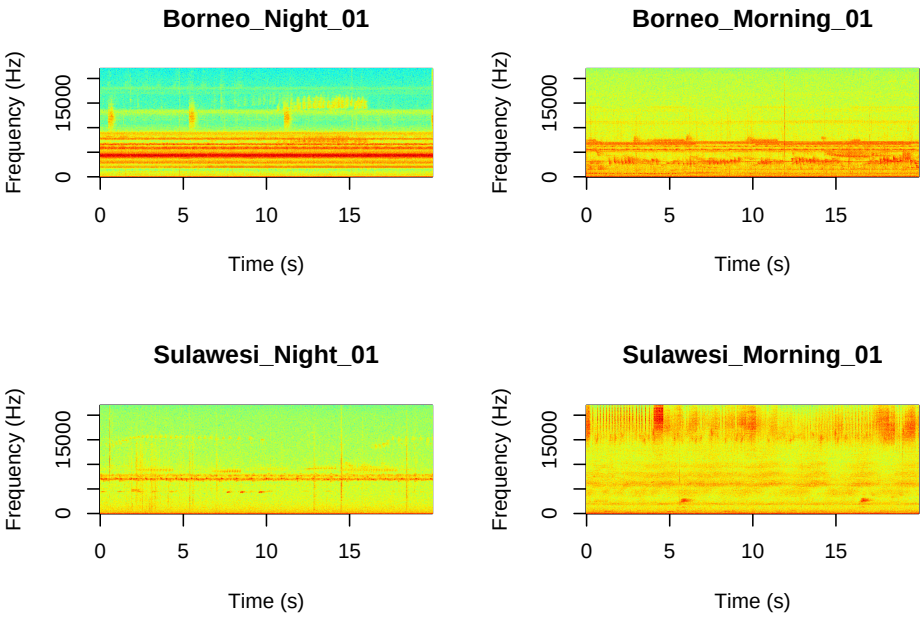
It looks like the Borneo_Night sampling period was much different from the rest! Let's look at a single spectrogram from each sampling location and time.

```
par(mfrow=c(2,2))
SpectrogramSingle(sound.file ="SoundscapeRecordings/Borneo_Night_01.wav",
                  min.freq = 0,max.freq=22000,
                  Colors = 'Colors',downsample = FALSE)

SpectrogramSingle(sound.file ="SoundscapeRecordings/Borneo_Morning_01.wav",
                  min.freq = 0,max.freq=22000,
                  Colors = 'Colors',downsample = FALSE)
```

```
SpectrogramSingle(sound.file ="SoundscapeRecordings/Sulawesi_Night_01.wav",
                  min.freq = 0,max.freq=22000,
                  Colors = 'Colors',downsample = FALSE)

SpectrogramSingle(sound.file ="SoundscapeRecordings/Sulawesi_Morning_01.wav",
                  min.freq = 0,max.freq=22000,
                  Colors = 'Colors',downsample = FALSE)
```



Question 3. You can listen to example sound files on the Canvas page. Between looking at the spectrograms and listening to the sound files what do you think are the main differences between the different locations and times?

Part 4. Now it is time to analyze the data you collected for this week's field lab.

Upload your sound files into either the 'MyFocalRecordings' folder or the 'MySoundscapeRecordings'folder and run the code below.

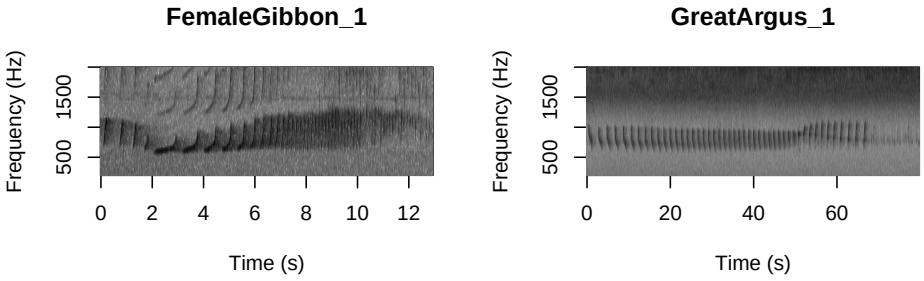
NOTE: Make sure to delete the existing files before you add your own!

3.0.1 Part 4a. Your focal recordings

First lets visualize the spectrograms. You may want to change the frequency settings depending on the frequency range of your focal animals.

```
# We can tell R to print the spectrograms 2x2 using the code below
par(mfrow=c(2,2))

# This is the function to create the spectrograms
SpectrogramFunction(input.dir = "MyFocalRecordings",min.freq = 200,max.freq=2000)
```



Here is the function to extract the features. Again, based on the frequency range of your focal recordings you may want to change the frequency settings.

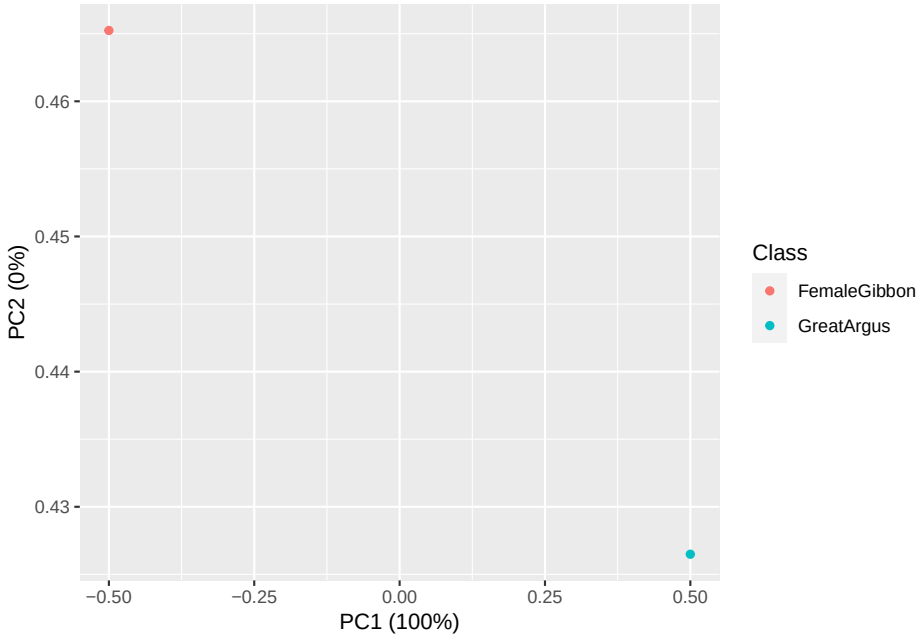
```
MyFeatureDataFrame <- MFCCFunction(input.dir = "MyFocalRecordings",min.freq = 200,max.freq=2000)
```

Then we run the principal component analysis

```
pca_res <- prcomp(MyFeatureDataFrame[,-c(1)], scale. = TRUE)
```

Now we visualize our results

```
ggplot2::autoplot(pca_res, data = MyFeatureDataFrame,
         colour = 'Class')
```

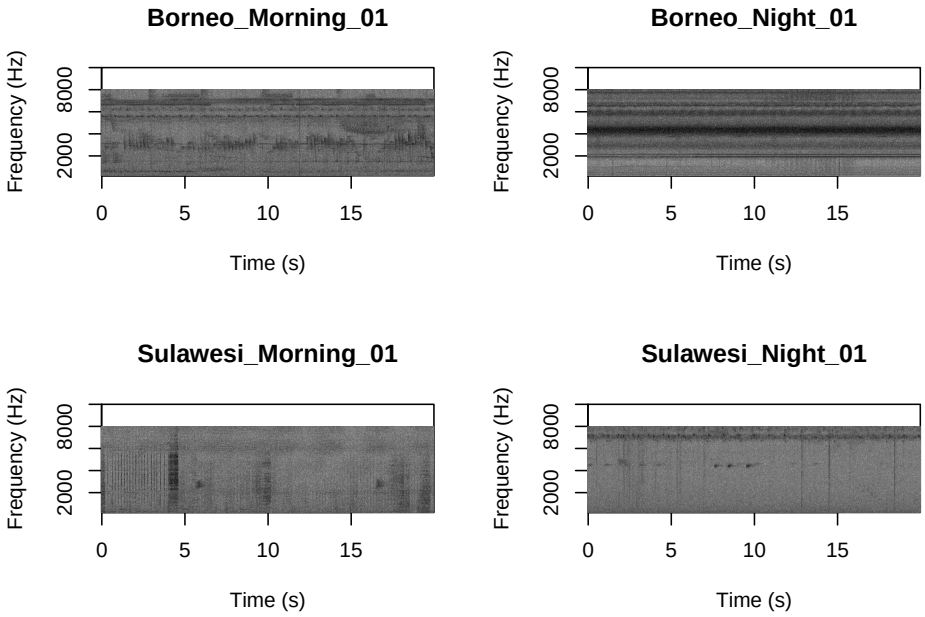


3.0.2 Part 4b. Soundscape recordings

First lets visualize the spectrograms. You probably don't want to change the frequency settings for this.

```
# We can tell R to print the spectrograms 2x2 using the code below
par(mfrow=c(2,2))

# This is the function to create the spectrograms
SpectrogramFunction(input.dir = "MySoundscapeRecordings",min.freq = 200,max.freq=10000)
```



Then we extract the features, which in this case are the MFCCs.

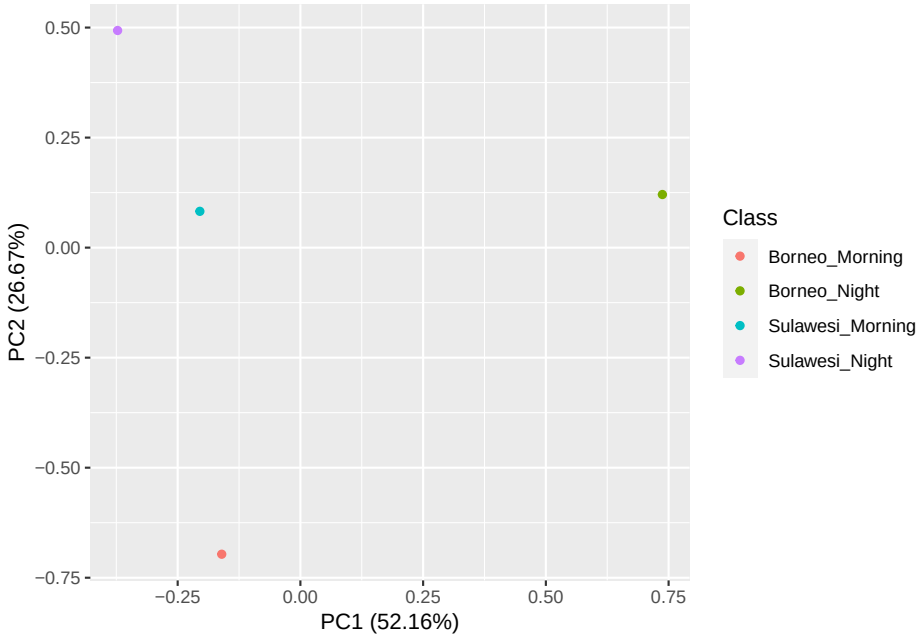
```
MySoundscapeFeatureDataframe <-
  MFCCFunctionSite(input.dir = "MySoundscapeRecordings",min.freq = 200,max.freq=10000)
```

Check the resulting structure of the dataframe to make sure it looks OK.

```
dim(MySoundscapeFeatureDataframe)
```

Now we visualize our results

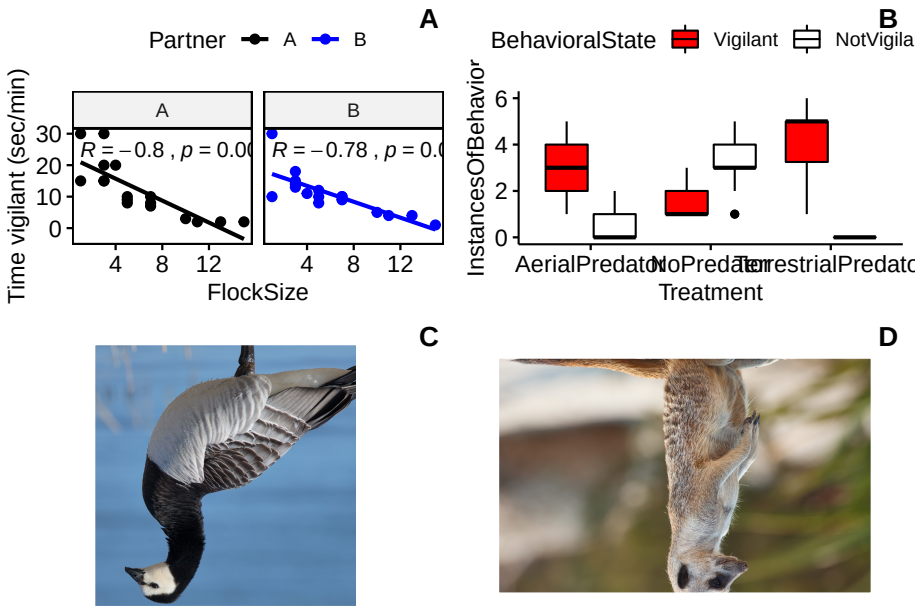
```
pca_res <- prcomp(MySoundscapeFeatureDataframe[,-c(1)], scale. = TRUE)
ggplot2::autoplot(pca_res, data = MySoundscapeFeatureDataframe,
         colour = 'Class')
```



Question 4. Do you see evidence of clustering in either your focal recordings or soundscape recordings?

Chapter 4

Lab 4. Vigilance behavior



Background
In this lab we will continue our work investigating vigilance behaviors in geese and meerkats.

Goals of the exercises
The main goal(s) of today's lab are to:
1) Use the data on goose vigilance behavior to investigate the relationship between group size and vigilance behaviors.
2) Analyze the data collected during the meerkat lab to test for differences in

vigilant behavior across predator treatment groups.
3) Continue to become familar with the way scientists analyze and interpret data.

Getting started
First we need to load the relevant packages for our data analysis. Packages contain all the functions that are needed for data analysis. First we load the required libraries

```
library(behaviouR)
```

4.1 Part 1: Barnacle goose vigilance

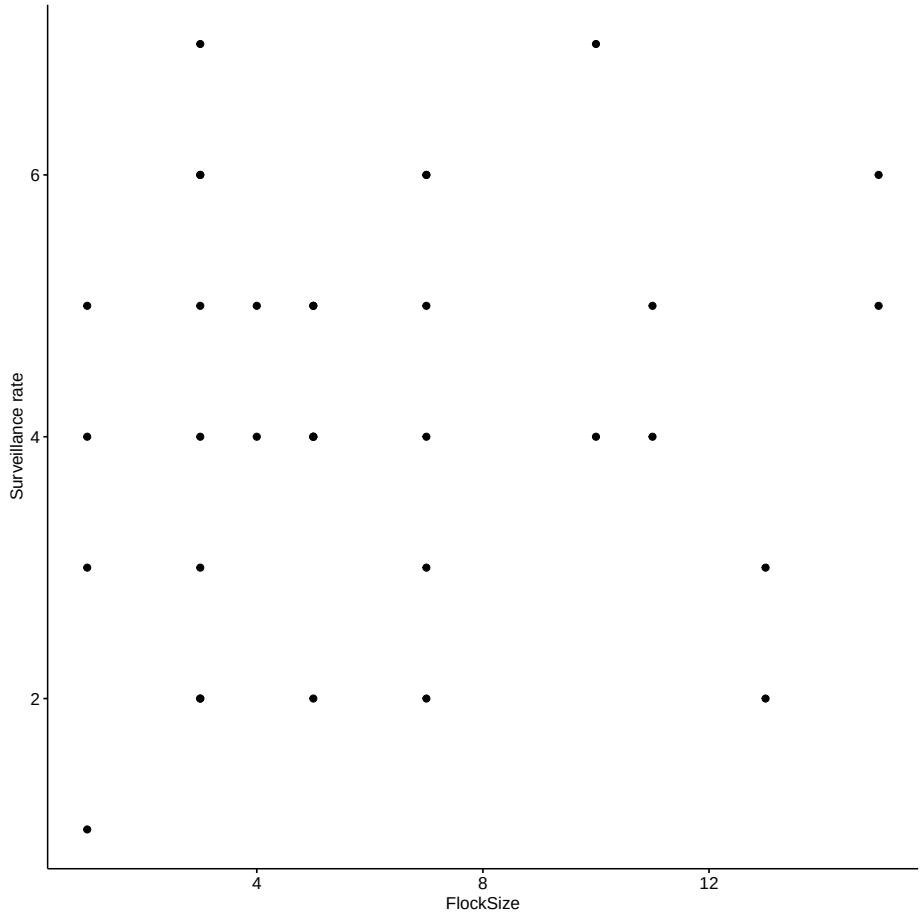
We load in our goose data

```
BarnacleGooseData <- read.csv('BarnacleGooseData.csv')
```

4.1.1 Part 1a: Surveillance behavior

First, we will calculate the surveillance rate (or the number of heads up per minute). Let's start by looking at the total number of 'head up' behaviors in our ethogram. To investigate this relationship we will create a scatterplot.

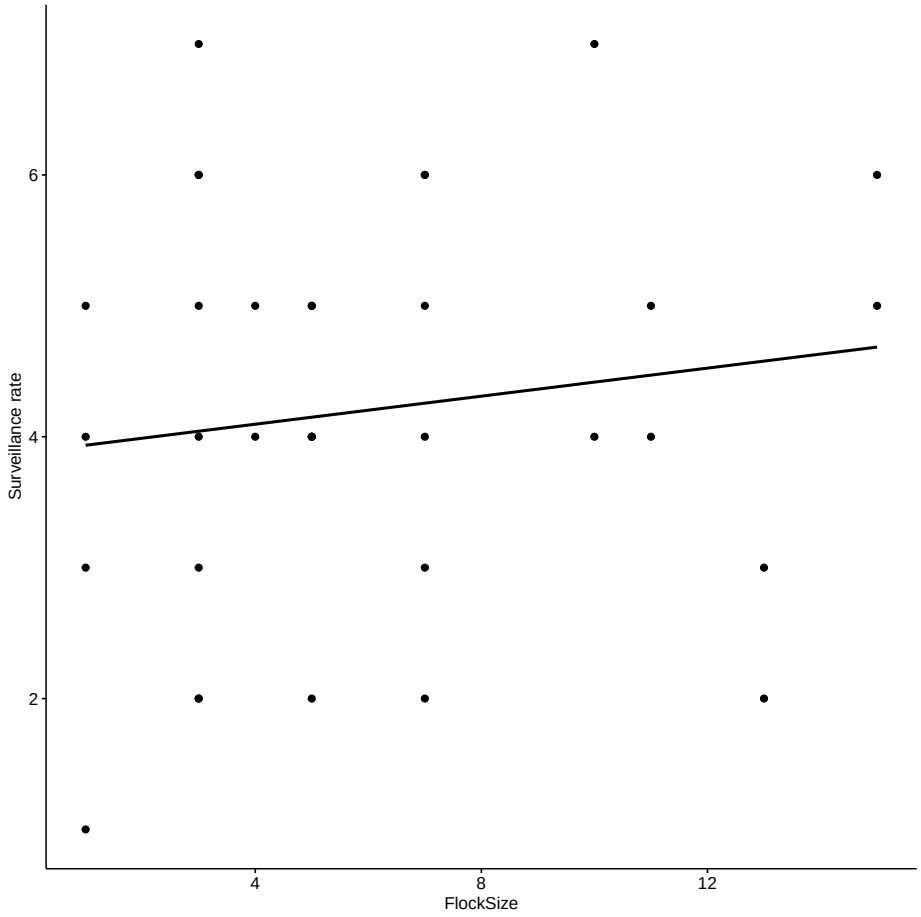
```
# Scatterplot of total number of 'head up' in our data
ggscatter(data=BarnacleGooseData,
          x='FlockSize',y='TotalHeadsUp')+ylab('Surveillance rate')
```



Now let's add a trend line to see if there is a relationship between flock size and the total number of 'head up' behaviors

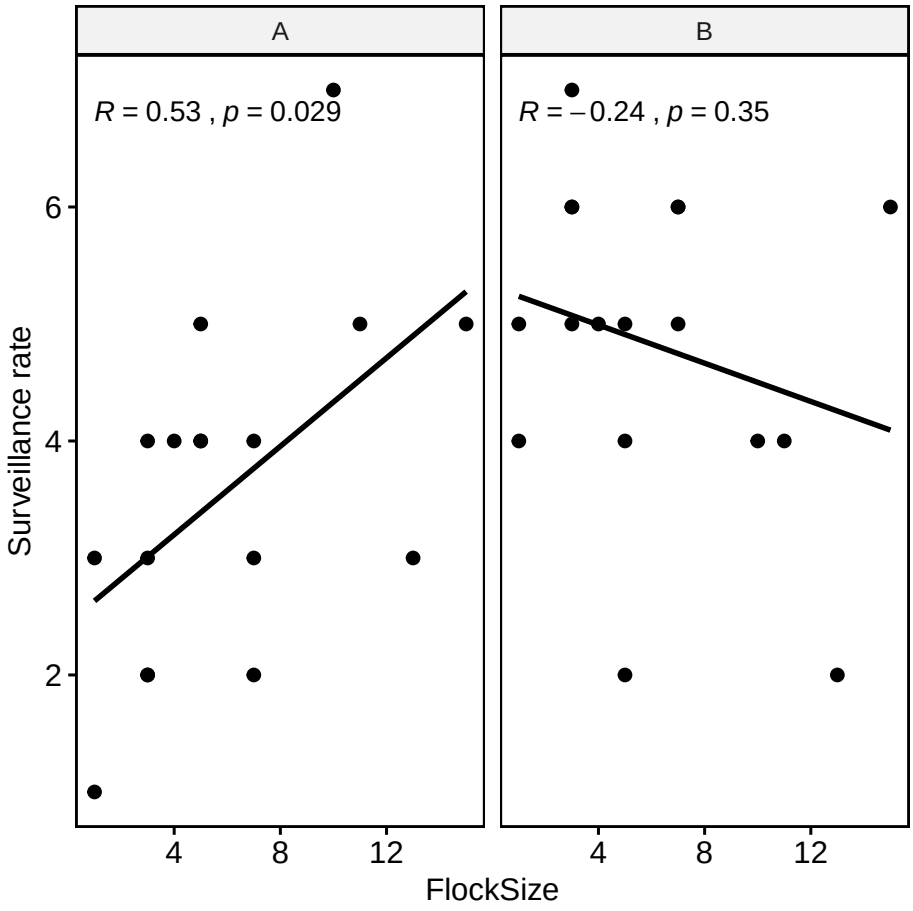
```
knitr::opts_chunk$set(fig.height = 5, fig.width = 5)
# Scatterplot of total number of 'head up' in our data with a trend line.

ggscatter(data=BarnacleGooseData,
          x='FlockSize',y='TotalHeadsUp',add='reg.line')+ylab('Surveillance rate')
```



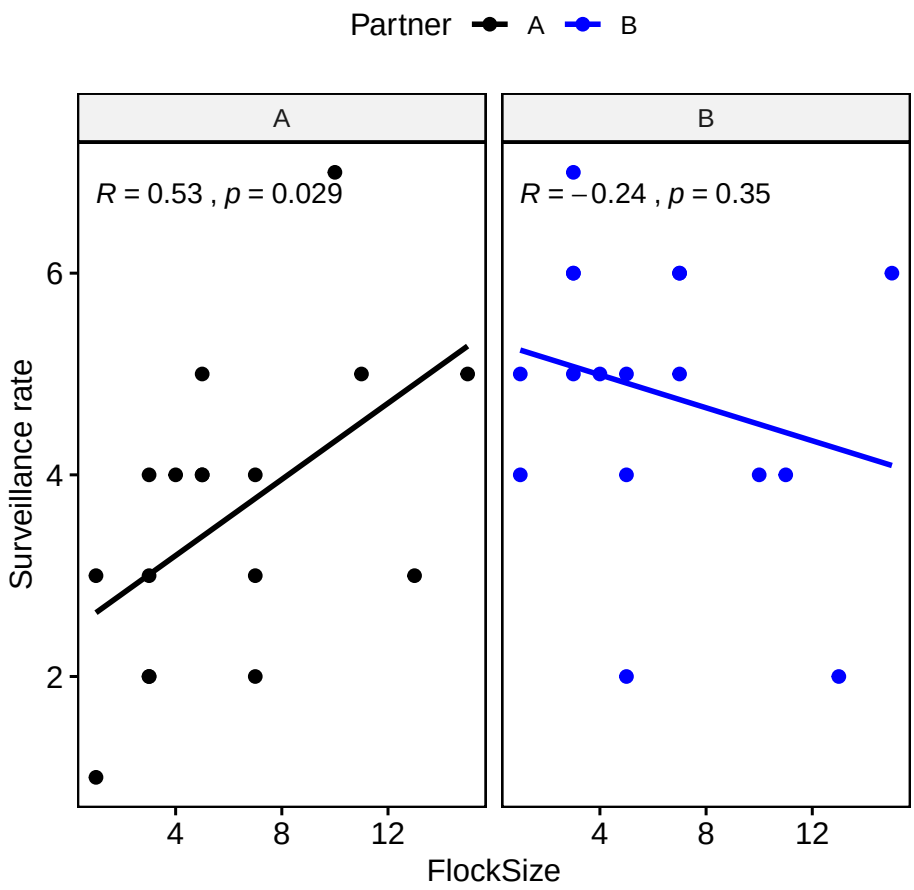
Let's see if there were any differences between you and your partner. We will also add the command 'cor.coef = T' which will give us the correlation coefficient (R) along with an associated p-value.

```
# NOTE: The data here are simulated so your plots should look different
ggscatter(data=BarnacleGooseData,
          x='FlockSize',y='TotalHeadsUp',add='reg.line', facet.by = 'Partner',
          cor.coef = T)+ylab('Surveillance rate')
```



Let's plot you and your partner's data in different colors.

```
ggscatter(data=BarnacleGooseData,
          x='FlockSize',y='TotalHeadsUp',add='reg.line', facet.by = 'Partner',
          color = 'Partner', palette =c('black','blue'),
          cor.coef = T)+ylab('Surveillance rate')
```



Question 1: Were there any major differences between you and your partner in terms of the observed relationship between flock size and surveillance rate?

Now we will do model selection using Akaike information criterion (AIC). First we create a null model and then we create a model with flock size as a predictor of total number of heads up.In the model code we specify that we are using a Poisson distribution, as we are dealing with count data instead of a continuous variable.

```
# This is our null model
SurveillanceNullModel <- glm(TotalHeadsUp ~ (1/Partner),family=poisson , data=Barnacle

# This is our model with flock size as a predictor of total number of heads up
SurveillanceModel <- glm(TotalHeadsUp ~ FlockSize + (1/Partner) ,family=poisson, data=

# Then we compare the models using AIC
bbmle::AICtab(SurveillanceNullModel,SurveillanceModel)
```

```
##                      dAIC df
```

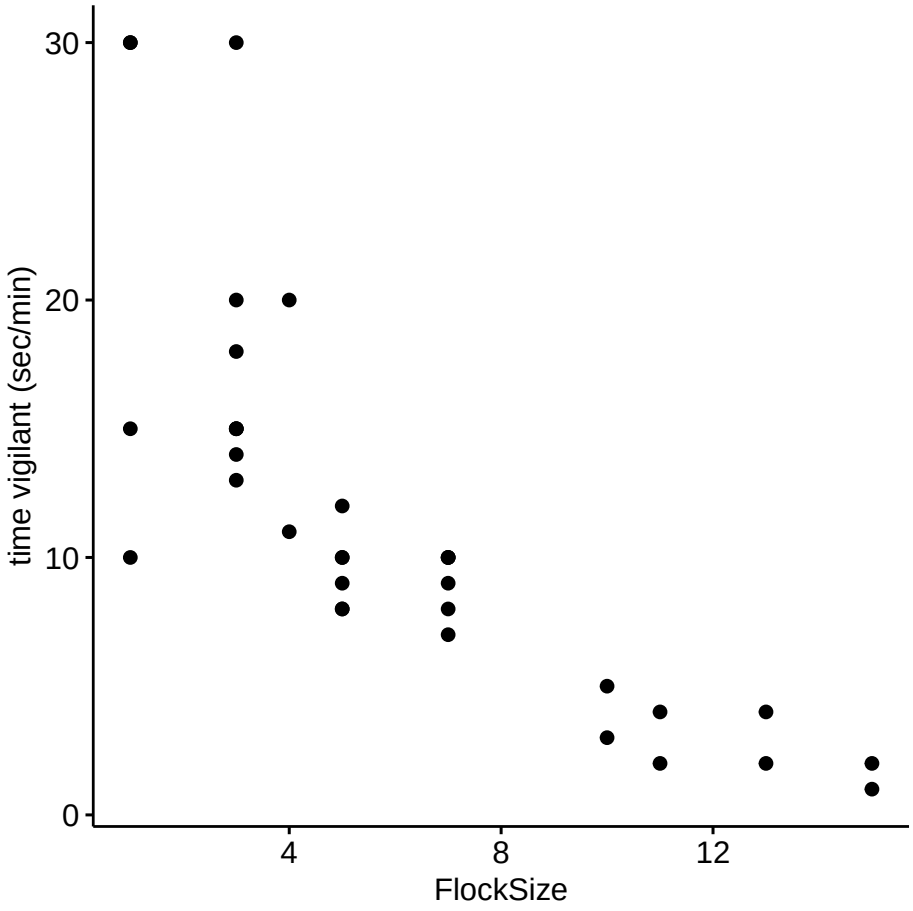
```
## SurveillanceNullModel 0.0  1
## SurveillanceModel     1.6  2
```

Question 2: If the null model is ranked higher than the model with flock size as a predictor, how do would we interpret this finding? What if the model with flock size as a predictor was ranked higher? What were your results?

4.1.2 Part 1b: Time vigilant (sec/min)

Now we will look at the relationship between the duration (calculated as seconds per minute) that the geese were vigilant as a function of flock size.

```
# Scatterplot of time vigilant (sec/min) as a function of group size
ggscatter(data=BarnacleGooseData,
          x='FlockSize',y='TimeSecHeadUp')+ylab('time vigilant (sec/min)')
```

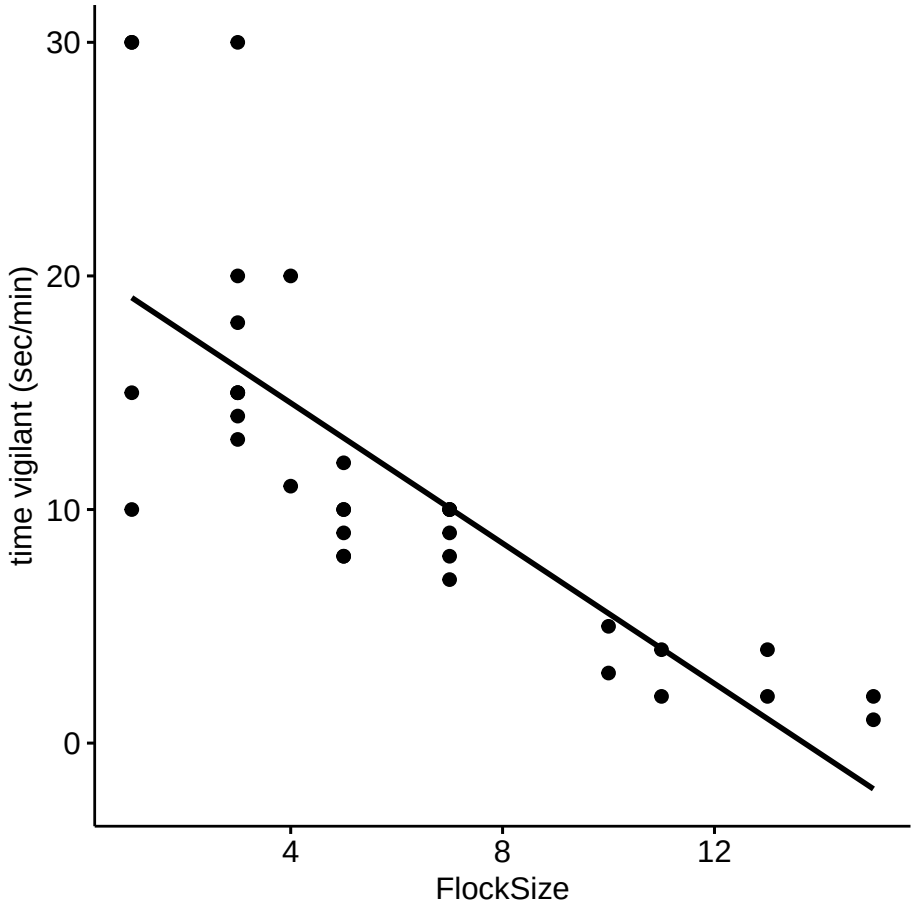


Now let's add a trend line to see if there is a relationship between flock size and

the duration of vigilance.

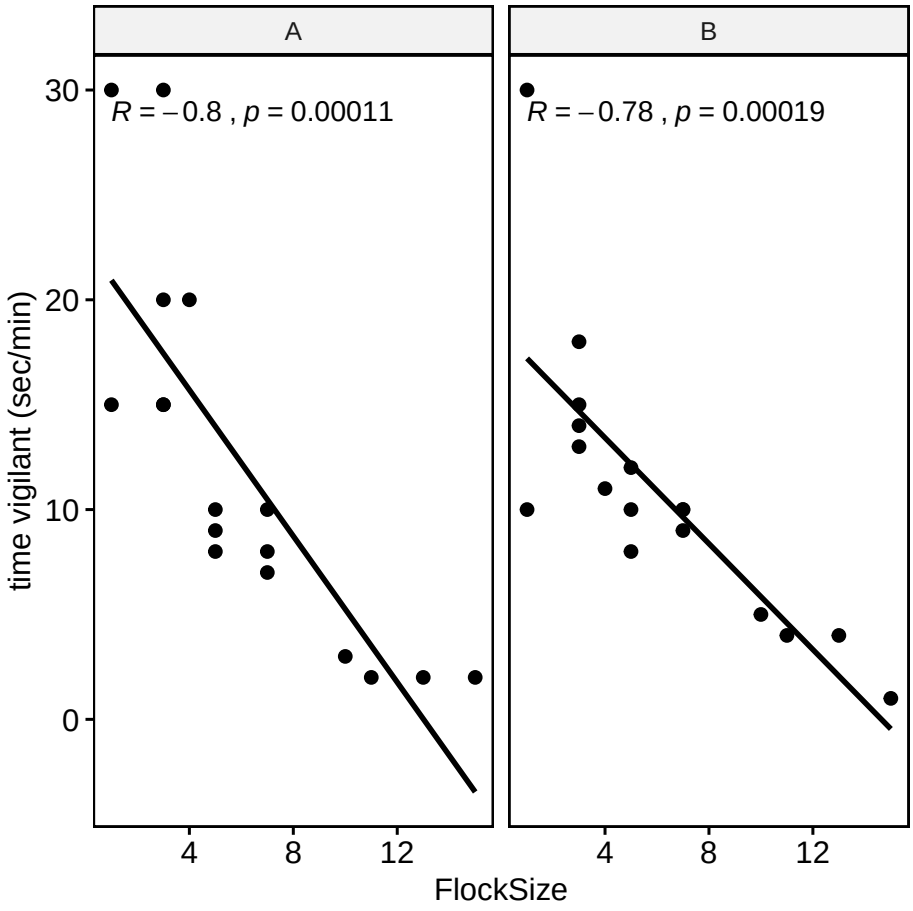
```
# Scatterplot of duration of vigilance behavior in our data with a trend line.

ggscatter(data=BarnacleGooseData,x='FlockSize',y='TimeSecHeadUp',add='reg.line')+
  ylab('time vigilant (sec/min)')
```



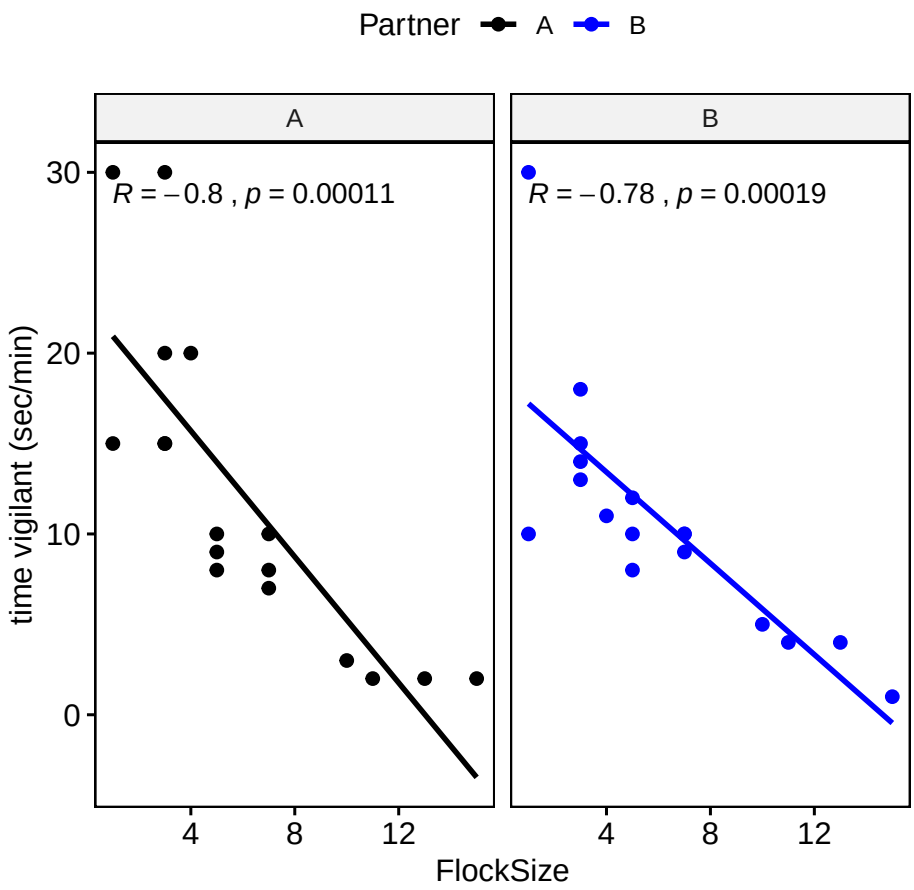
Let's see if there were any differences between you and your partner. We will also add the command 'cor.coef = T' which will give us the correlation coefficient (R) along with an associated p-value.

```
# NOTE: The data here are simulated so your plots should look different
ggscatter(data=BarnacleGooseData,x='FlockSize',y='TimeSecHeadUp',add='reg.line',
          facet.by = 'Partner',cor.coef = T)+
  ylab('time vigilant (sec/min)')
```



Let's plot you and your partner's data in different colors.

```
ggscatter(data=BarnacleGooseData,x='FlockSize',y='TimeSecHeadUp',add='reg.line',
          facet.by = 'Partner', color='Partner', cor.coef = T,
          palette =c('black','blue'))+
            ylab('time vigilant (sec/min)')
```



As before we will create a null model and then a model with flock size as a predictor and compare them using AIC. We will not use a Poisson distribution here because our outcome variable is continuous.

```
# This is our null model
VigilanceNullModel <- lme4::lmer(TimeSecHeadUp ~ (1|Partner), data=BarnacleGooseData)

# This is our model with flock size as a predictor duration of vigilance
VigilanceModel <- lme4::lmer(TimeSecHeadUp ~ FlockSize + (1|Partner) ,data=BarnacleGoos

# Now we compare the models using AIC
bbmle::AICtab(VigilanceNullModel,VigilanceModel)
```

```
##                    dAIC df
## VigilanceModel      0.0 4
## VigilanceNullModel 28.1 3
```

#**Question 3.** How do you interpret the results of your model selection? Was there a relationship between flock size and duration of vigilance behavior?

4.2 Part 2: Meerkat data revisted

Please upload your meerkat scan data to this project and delete the existing datasheet.

```
MeerkatScanData <- read.csv('MeerkatScanData.csv')
```

As before we will turn our NA values to zero

```
MeerkatScanData[is.na(MeerkatScanData)] <- '0'
```

We will remove the time and out of sight columns as we do not need them

```
MeerkatScanData <- dplyr::select(MeerkatScanData,-c(Time,OutOfSight))
```

We need to reformat our data so that we can plot it

```
MeerkatScanDataSummaryLong <- reshape2::melt(MeerkatScanData, id.vars=c("Treatment", "Partner"))
```

Here we add more informative column names

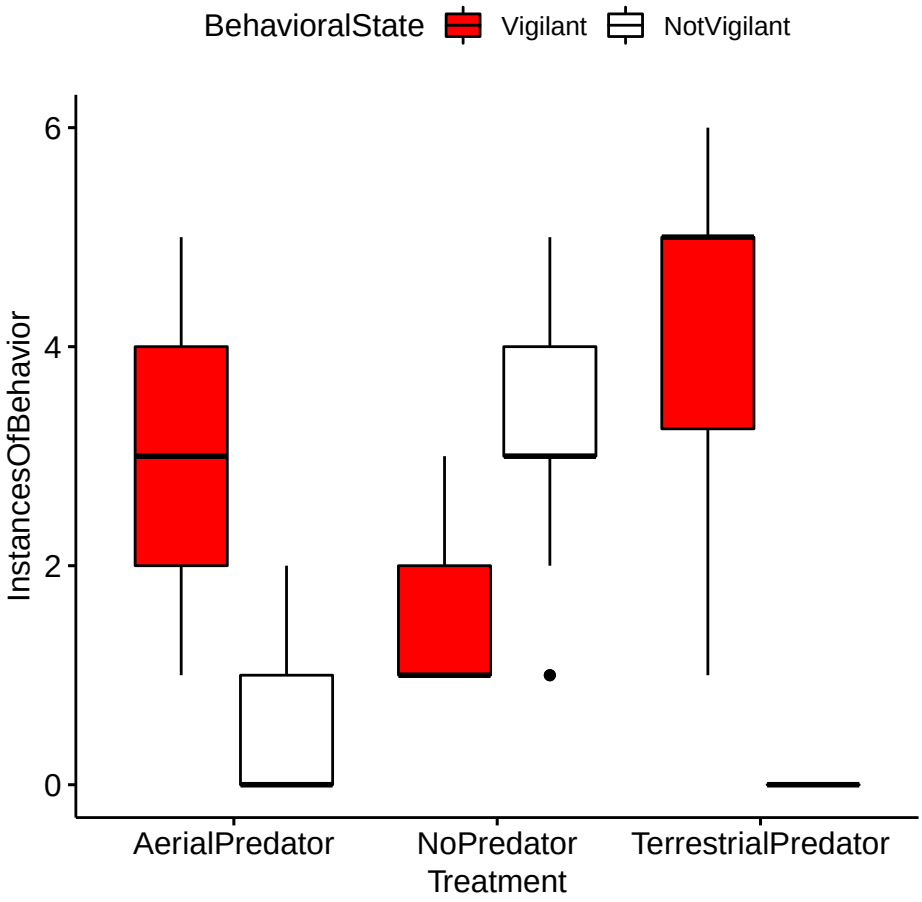
```
colnames(MeerkatScanDataSummaryLong) <- c('Treatment','Partner',
                                          'BehavioralState','InstancesOfBehavior')
```

We need to tell R that our outcome variable is not categorical but numeric

```
MeerkatScanDataSummaryLong$InstancesOfBehavior <-
  as.numeric(MeerkatScanDataSummaryLong$InstancesOfBehavior)
```

Now we plot our data.

```
ggboxplot(MeerkatScanDataSummaryLong,x='Treatment',
          y='InstancesOfBehavior', fill = 'BehavioralState')+scale_fill_manual(values = c('red','
```



Question 4. Based on your inspection of the boxplots, are there any major differences between treatment groups?

Now we will test to see if there were differences in vigilance behaviors across treatments.

```
# First we subset our data so that it only includes the vigilant category
MeerkatScanDataVigilantOnly <- subset(MeerkatScanDataSummaryLong,
                                      BehavioralState=='Vigilant' )

# R can be picky about the format of data, so we use this command to tell R that treat
MeerkatScanDataVigilantOnly$Treatment <-
  as.factor(MeerkatScanDataVigilantOnly$Treatment)

# Here we are reordering the levels of the factors. For our model selection we are int

MeerkatScanDataVigilantOnly$Treatment <-
  factor(MeerkatScanDataVigilantOnly$Treatment, levels = c("NoPredator", "AerialPredat
```

```
                                                    "TerrestrialPredator"))
```

Now as before we will do model selection. Note that because our outcome variable (instances of behavior) is in the form of count data we use a poisson distribution.

```
# This is the null model.
MeerkatVigilanceNullModel <- glm(InstancesOfBehavior ~ 1, family=poisson, data=MeerkatScanDataVig

# This is the model with treatment as a predictor of instances of vigilant behavior
MeerkatVigilanceModel <- glm(InstancesOfBehavior ~ Treatment, family=poisson,data=MeerkatScanData
```

Now we compare the models using AIC

```
bbmle::AICtab(MeerkatVigilanceNullModel,MeerkatVigilanceModel)
```

```
##                            dAIC df
## MeerkatVigilanceModel       0.0 3
## MeerkatVigilanceNullModel 59.2 1
```

Here we will use the summary function to look at the estimates. There is a lot of information here but we want to focus on the 'Estimate'. In particular we are interested in the estimates for 'TreatmentAerialPredator' and 'TreatmentTerrestrialPredator'. The estimate is showing the effect that these variables have on our outcome (instances of behavior), relative to our control (no predator).Therefore positive estimates indicate that there were more vigilance behaviors in aerial and terrestrial predator treatments.

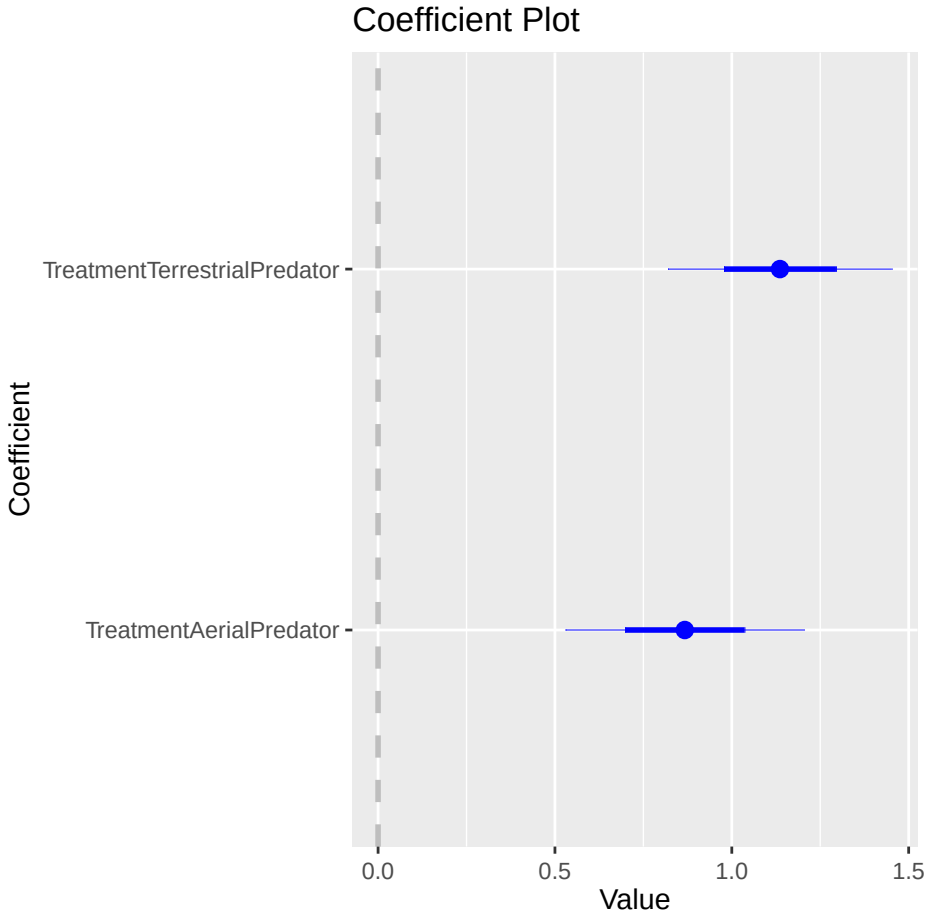
```
summary(MeerkatVigilanceModel)
```

```
##
## Call:
## glm(formula = InstancesOfBehavior ~ Treatment, family = poisson,
##     data = MeerkatScanDataVigilantOnly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9295  -0.3475  -0.1588   0.3696   1.1825
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    0.3285     0.1414   2.323   0.0202 *
## TreatmentAerialPredator        0.8671     0.1685   5.145 2.68e-07 ***
## TreatmentTerrestrialPredator   1.1362     0.1582   7.182 6.86e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
```

```
##
##     Null deviance: 100.802  on 117  degrees of freedom
## Residual deviance:  37.587  on 115  degrees of freedom
## AIC: 384.39
##
## Number of Fisher Scoring iterations: 4
```

A common way to visulize results such as these are coefficient plots. Here we are looking at the effect of 'TreatmentAerialPredator' and 'TreatmentTerrestrialPredator' relative to our control group. The reference or group is indicated by the vertical dashed line. So, we can interpret that because the coefficients are positive (and the confidence intervals don't overlap zero) that both terrestrial and aerial treatments lead to an increase in vigilant behaviors.

```
coefplot::coefplot(MeerkatVigilanceModel,intercept=F)
```



For reasons that will not go into here, I am not a fan of p-values or null hypothesis significance testing. There is a nice overview if you want to learn more here:

https://doi.org/10.1098/rsbl.2019.0174. But,the model selection approach we used will lead to the same inference as the use of a one-way anova, an approach that you may be more familiar with.

Compute the analysis of variance

```
MeerkatAOV <- aov(InstancesOfBehavior ~ Treatment, data = MeerkatScanDataVigilantOnly)
```

Here this will tell us if there are differences between groups. A significant p-value (< 0.05) indicates there are differences between treatment groups.

```
summary(MeerkatAOV)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Treatment     2  176.0   88.02   87.03 <2e-16 ***
## Residuals   115  116.3    1.01
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the ANOVA test is significant, we can compute Tukey Honest Significant Differences test. Again, a significant p-value (< 0.05) indicates there are differences between treatments.

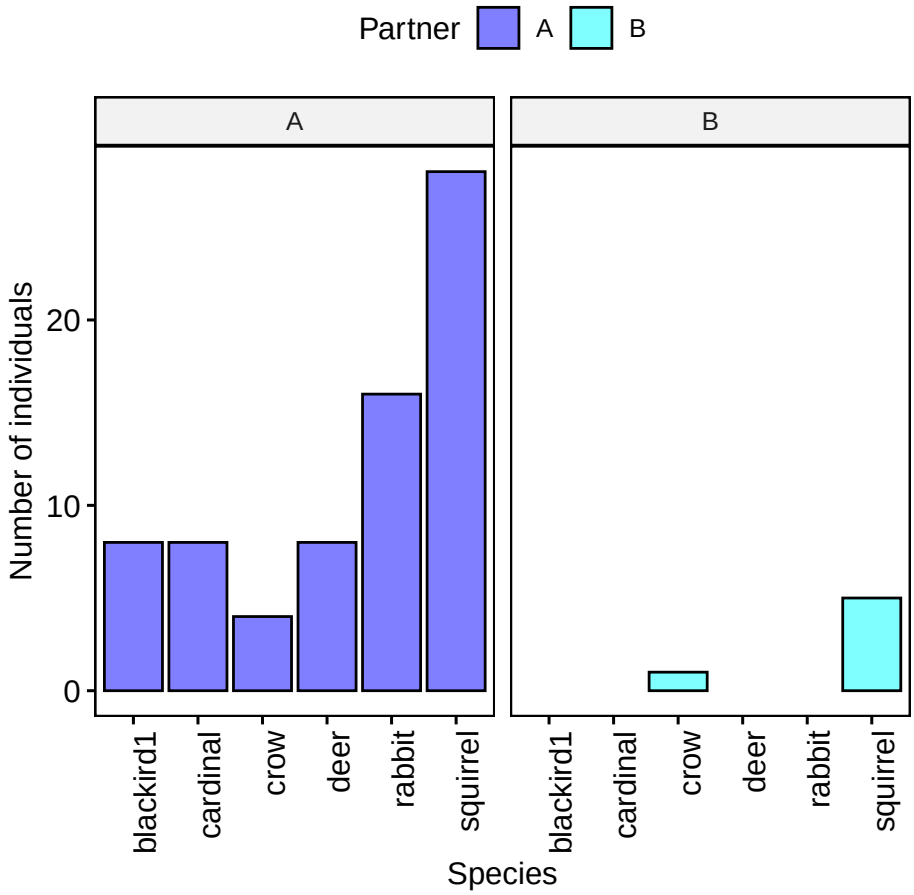
```
TukeyHSD(MeerkatAOV)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = InstancesOfBehavior ~ Treatment, data = MeerkatScanDataVigilantOnly)
##
## $Treatment
##                                         diff       lwr      upr   p adj
## AerialPredator-NoPredator           1.916667 1.3538432 2.479490 0.0e+00
## TerrestrialPredator-NoPredator      2.937198 2.4058425 3.468554 0.0e+00
## TerrestrialPredator-AerialPredator 1.020531 0.4891758 1.551887 3.8e-05
```

Question 5. Based on your interpretation of the model selection and the coefficient plots were there differences between treatment groups (e.g. control, terrestrial and aerial predators) in meerkat vigilance behavior?

Chapter 5

Lab 5. Estimating Population Density and Biodiversity



Background
In this lab you will become familiar with the ways that scientists estimate population density and biodiversity of terrestrial vertebrates.

Goals of the exercises
The main goal(s) of today's lab are to:
1) Estimate population density of focal vertebrates from the field lab.
2) Learn the difference between alpha, beta and gamma diversity.
3) Compare population density and biodiversity across your sampling sites.

Getting started
First we need to load the relevant packages for our data analysis. Packages contain all the functions that are needed for data analysis.

```
library(behaviouR)
```

Then we read in our data

```
CensusData <- read.csv('FieldLab5CensusData.csv')
```

5.1 Part 1. Population density estimation.

First let's focus on your data.
NOTE: you need to change 'A' to the name you used!

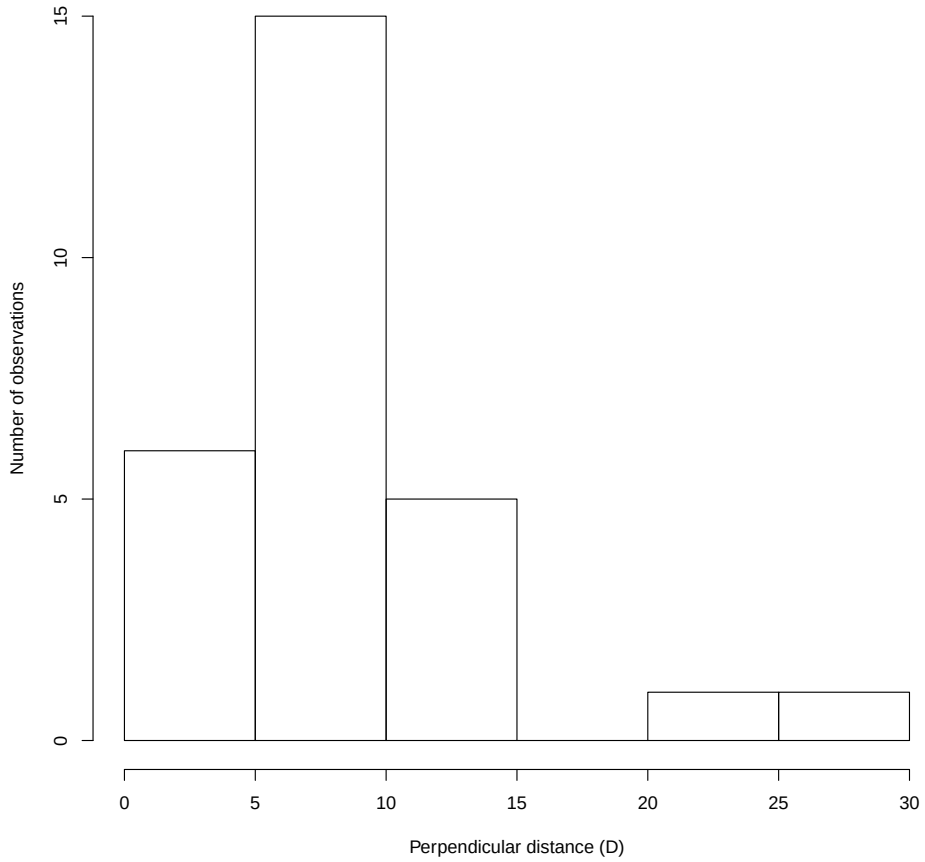
```
MyCensusData <- subset(CensusData, Partner=='A')
```

Now we will subset the data to focus on your focal species for density estimation.
NOTE: in this example we are subsetting the data so that we only have the squirrels; you will need to change the code to subset based on your focal species!

```
MyCensusDataFocal <- subset(MyCensusData, Species=='squirrel')
```

Here we will determine the width of our sampling area by creating a histogram of perpendicular detection distances. We will identify the point where our detections start to dropoff. We will assume that after this point our ability to detect animals past that distance drops substantially, and including observations past this distance could potentially bias our results.

```
hist(MyCensusDataFocal$PerpendicularDistance,xlab='Perpendicular distance (D)',
  ylab='Number of observations',main='')
```



In this example there is a clear break between 15 and 20 meters, so we will only use observations that were within 15 meters.
NOTE: Your data will look different than this!

Change the value (in this example 15) to the cutoff point indicated in your data!

```
CutOffPoint <- 15
```

```
# Here we subset our data so that it only includes detections that were within 15 mete
MyCensusDataFocalAdjusted <- subset(MyCensusDataFocal,PerpendicularDistance <CutOffPoi
```

Now we need to subset by each site.
NOTE: You will need to change to the site names you used

```
MyCensusDataFocalSiteA <- subset(MyCensusDataFocalAdjusted,Site=='SiteA')

MyCensusDataFocalSiteB <- subset(MyCensusDataFocalAdjusted,Site=='SiteB')
```

Now we will calculate the population density based on our two surveys. Change the following to indicate the distance (in meters) of your survey for site A. This will be the actual straight-line distance you walked.

```
SiteACensusDistance <- 500
```

Now we will calculate the area of our census. The sample area (a) is equal to the length of the transect multiplied by twice the width or a= 2wl. We divide by 1000 to convert our answer to square kilometers.

```
SiteACensusArea <- 2*SiteACensusDistance*CutOffPoint/1000
SiteACensusArea
```

```
## [1] 15
```

Now we need to calculate the number of focal animals observed using the following code.

```
NumberFocalAnimalsSiteA <- nrow(MyCensusDataFocalSiteA)
```

Then we can calculate the density by dividing the total number of animals we observed by the area we censused.

```
PopulationDensitySiteA <-NumberFocalAnimalsSiteA/SiteACensusArea
PopulationDensitySiteA
```

```
## [1] 0.4666667
```

Now we will calculate the population density based on our second survey. Change the following to indicate the distance (in meters) of your survey for site B.This will be the actual straight-line distance you walked.

```
SiteBCensusDistance <- 500
```

Now we will calculate the area of our census. The sample area (a) is equal to the length of the transect multiplied by twice the width or a= 2wl.We divide by 1000 to convert our answer to square kilometers.

```
SiteBCensusArea <- 2*SiteBCensusDistance*CutOffPoint/1000
SiteBCensusArea
```

```
## [1] 15
```

Now we need to calculate the number of animals using the following code.

```
NumberFocalAnimalsSiteB <- nrow(MyCensusDataFocalSiteB)
```

Then we can calculate the density by dividing the total number of animals we observed by the area we censused.

```
PopulationDensitySiteB <-NumberFocalAnimalsSiteB/SiteBCensusArea
PopulationDensitySiteB
```

```
## [1] 1.133333
```

Now we can compare population density using the following code. The code below asks if the population density of site A was higher than site B?

```
PopulationDensitySiteA > PopulationDensitySiteB
```

```
## [1] FALSE
```

The code below asks if the population density of site B was higher than site A?

```
PopulationDensitySiteB > PopulationDensitySiteA
```

```
## [1] TRUE
```

Question 1. What were the population density estimates (reported as number of individuals per square kilometer) for your two sites? Do your results of the population density estimates match your predictions? Why or why not?

5.2 Part 2. Comparing biodiversity.

5.2.1 Alpha diversity.

First, we will estimate the alpha diversity, or the diversity within a particular area or ecosystem. The alpha diversity is simply the number of different species present at each site.

Here we subset by partner and site A (you may need to change these!)

```
MyCensusDataSiteA <- subset(MyCensusData,Partner=='A' & Site=='SiteA')

# What were the unique species present?
unique(MyCensusDataSiteA$Species)
```

```
## [1] cardinal  blackird1 rabbit    deer      squirrel  crow
## Levels: blackird1 cardinal crow deer rabbit squirrel
```

```
# How many unique species were there?
SiteANumberSpecies <- length(unique(MyCensusDataSiteA$Species))
SiteANumberSpecies
```

```
## [1] 6
```

Here we subset by partner and site B (you may need to change these!)

```
MyCensusDataSiteB <- subset(MyCensusData,Partner=='A' & Site=='SiteB')

# What were the unique species present?
unique(MyCensusDataSiteB$Species)
```

```
## [1] squirrel  blackird1 rabbit    deer      crow
## Levels: blackird1 cardinal crow deer rabbit squirrel
```

```
# How many unique species were there?
SiteBNumberSpecies <- length(unique(MyCensusDataSiteB$Species))
SiteBNumberSpecies
```

```
## [1] 5
```

Question 2. Which of your sites had higher species richness (i.e. number of species)?

5.2.2 Beta diversity.

Now we will estimate beta diversity, which estimates changes in species diversity between ecosystems or along environmental gradients.

```
# This code tells us which species both sites have in common
intersect(unique(MyCensusDataSiteA$Species),unique(MyCensusDataSiteB$Species))
```

```
## [1] "blackird1" "rabbit"    "deer"      "squirrel"  "crow"
```

```
# Now we calculate the number of species in common
SpeciesInCommonBothSites <- length(intersect(unique(MyCensusDataSiteA$Species),unique(MyCensusDat
)
SpeciesInCommonBothSites
```

```
## [1] 5
```

To investigate community similarity we will calculate Sørenson's index; a value of 1 means exactly the same number of species a value of 0 means no overlap.

Beta diversity = 2c / S1 + S2
Where c is the number of species the sites have in common, S1 is the number of species at the first site and S2 is the number of species at the second site

```
2*SpeciesInCommonBothSites/ (SiteANumberSpecies+SiteBNumberSpecies)
```

```
## [1] 0.9090909
```

5.2.3 Now we will use your partner's data to estimate alpha diversity

```
MyPartnersCensusDataSiteA <- subset(CensusData,Partner=='B' & Site=='SiteC')
unique(MyPartnersCensusDataSiteA$Species)
```

```
## [1] squirrel crow
## Levels: blackird1 cardinal crow deer rabbit squirrel
```

```
length(unique(MyPartnersCensusDataSiteA$Species))
```

```
## [1] 2
```

```
SiteANumberSpecies <- length(unique(MyPartnersCensusDataSiteA$Species))
SiteANumberSpecies
```

```
## [1] 2
```

```
MyPartnersCensusDataSiteB <- subset(CensusData,Partner=='B' & Site=='SiteD')
unique(MyPartnersCensusDataSiteB$Species)
```

```
## [1] squirrel
## Levels: blackird1 cardinal crow deer rabbit squirrel
```

```
length(unique(MyPartnersCensusDataSiteB$Species))
```

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

```
SiteBNumberSpecies <- length(unique(MyPartnersCensusDataSiteB$Species))
SiteBNumberSpecies
```

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

Question 3. How did the alpha diversity of each of your sites compare with that of your partner?

5.2.4 Gamma diversity

Gamma diversity is the total number of species over a large area or region; there are many different ways that this can be measured. The way we will do it is a bit of an oversimplification by simply comparing the number of species seen during the census at both locations.

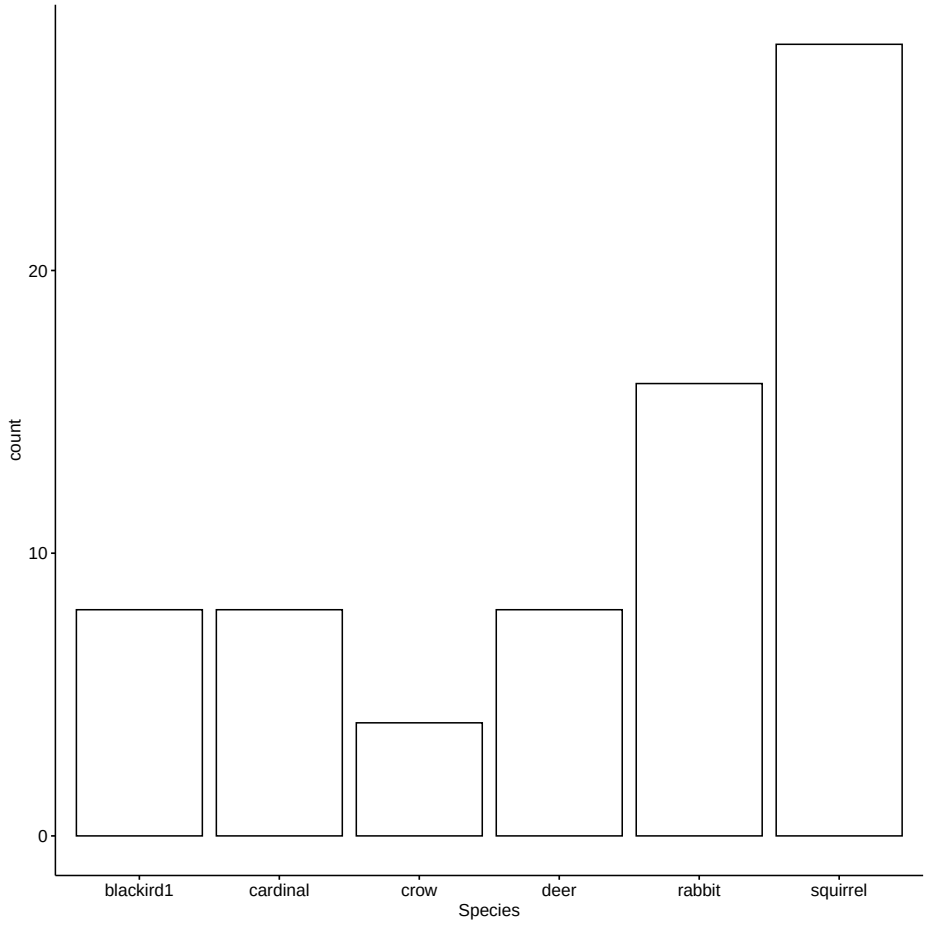
First we subset by the first partner

```
MyCensusData <- subset(CensusData, Partner=='A')
unique(MyCensusData$Species)
```

```
## [1] cardinal  blackird1 rabbit    deer      squirrel  crow
## Levels: blackird1 cardinal crow deer rabbit squirrel
```

Then we plot the results

```
gghistogram(data=MyCensusData, x='Species',stat="count")
```



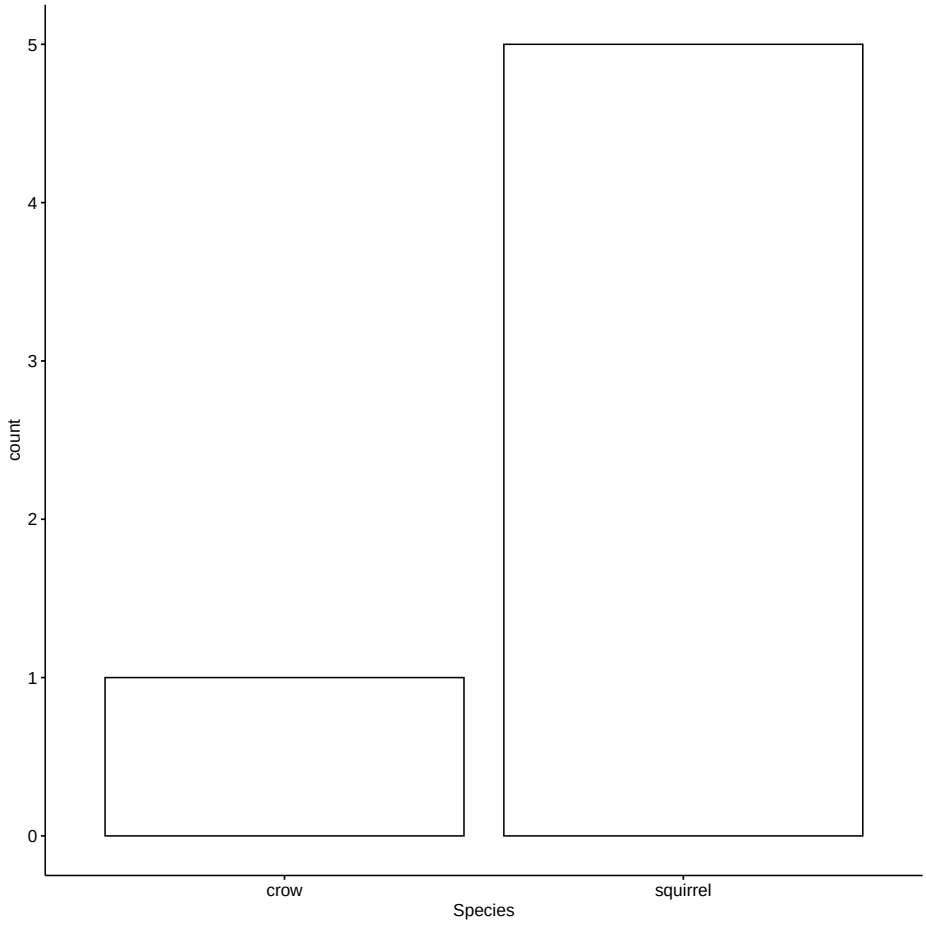
Now we subset by the second partner

```
MyPartnersCensusData <- subset(CensusData, Partner=='B')
unique(MyPartnersCensusData$Species)
```

```
## [1] squirrel crow
## Levels: blackird1 cardinal crow deer rabbit squirrel
```

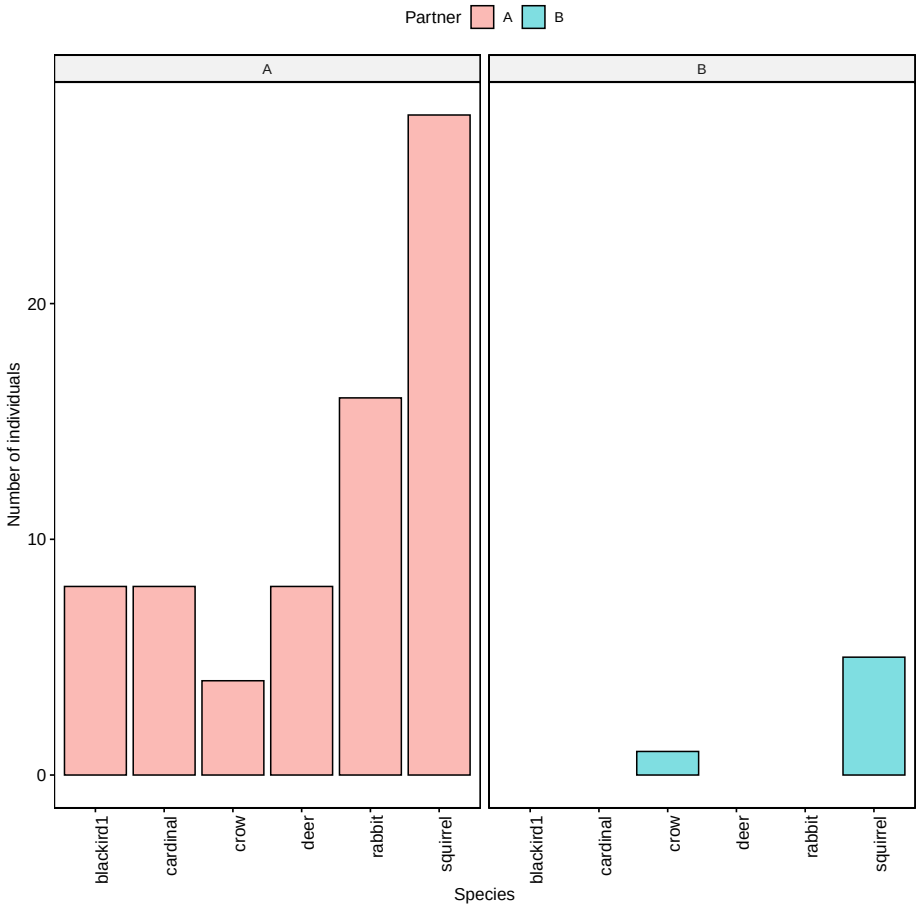
Then we plot the results

```
gghistogram(data=MyPartnersCensusData, x='Species',stat="count")
```



Now we can plot all the data together, separated by partner

```
gghistogram(data=CensusData, x='Species',stat="count",
            facet.by = 'Partner',x.text.angle =90,
            fill='Partner')+xlab('Species')+ylab('Number of individuals')
```



Question 4. How did the gamma diversity of your site compare with that of your partners?

5.3 Part 3. Biodiversity indices in the real world.

There are special packages in R that can measure different diversity indices, as biodiversity indices are tools many ecologists use. The package we will use is called 'vegan'.

```
library(vegan)
```

First we need to convert our data into a table that can be used to calculate the indices.

```
BiodiversityTable <- table(CensusData$Site,CensusData$Species)
```

Simpson's Index (D) measures the probability that two individuals randomly selected from a sample will belong to the same species (or some category other than species). With this index, 1 represents infinite diversity and 0 means no diversity.

```
H <- diversity(BiodiversityTable,index="simpson")
H
```

```
##     SiteA     SiteB     SiteC     SiteD
## 0.7962963 0.6728395 0.4444444 0.0000000
```

Species evenness refers to how close in numbers each species is in an environment. We can calculate evenness using the following code. The value is constrained between 0 and 1, with communities that have a more even representation of species having values closer to 1.

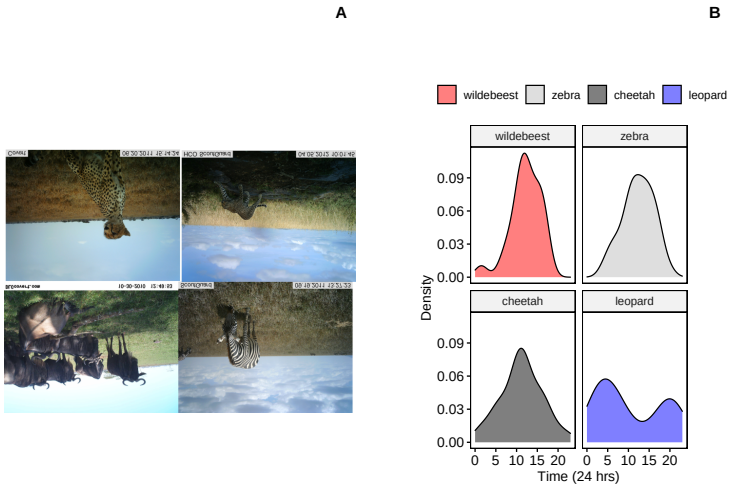
```
J <- H/log(specnumber(BiodiversityTable))
J
```

```
##     SiteA     SiteB     SiteC     SiteD
## 0.4444214 0.4180587 0.6411978       NaN
```

Question 5. Which of the four sites you analyzed was most diverse? Which was the most even? Why is it important to consider diversity and evenness when studying biodiversity?

Chapter 6

Lab 6. Analyzing camera trap data.



Background
In this lab we will use actual camera trap data collected in Serengeti National Park to investigate temporal niche partitioning. The camera trap data that we will use for this field lab comes from the 'Snapshot Serengeti' project (http://lila.science/datasets/snapshot-serengeti). The scientists set out out 225 cameras within a 1,125 km2 area. The cameras have been deployed continuously since 2010, and the researchers recruited citizen scientists to help with classifying images (https://www.zooniverse.org/projects/zooniverse/snapshot-serengeti).

Goals of the exercises
The main goal(s) of today's lab are to:

1) Understand the types of behavioral data that scientists can collect from camera trap photos.
2) Become familiar with one of the ways scientists can compare differences in activity patterns.
3) Think about interspecific interactions of predator-prey and potential competitors in the Serengeti.

Getting started
First we need to load the relevant packages for our data analysis. Packages contain all the functions that are needed for data analysis.

```
library(behaviouR)
```

6.1 Part 1: Collect Serengeti camera trap data

The original dataset includes over 2.65M sequences of camera trap images (totaling 7.1M images) from ten field seasons. As you can imagine that is a lot of data! For this lab we will focus on a subset of the data collected over the course of a few seasons.

We will be taking random subsets of the camera trap photos (the current default is 5 per season which you should change for your actual study), and we use 'set.seed' to make sure that our results are reproducible.

```
set.seed(2210)
```

We can use the following code to query the database by season to see which photos are available. The available seasons are 1-4, 6-8,10,11. This function will return a table with all of the available camera trap photos for a particular season. With camera trap data often times photos will be taken in a sequence (e.g. if the animal is moving in front of the camera for a long time). The values in the table below include all photos in a sequence, but the code we use to download the photos only takes the first photo in a sequence. Therefore the sample sizes indicated in this table and the actual sample size may be different.

```
CameraTrapAnnotations(season = 1)
```

```
##
##        aardvark        aardwolf          baboon     batearedfox         buffalo
##             143              88             348             315            1455
##        bushbuck         caracal         cheetah           civet         dikdik
##              36              57             575              27             728
##           eland        elephant   gazellegrants gazellethomsons          genet
##             113            1687            4967           29705              28
##         giraffe      guineafowl            hare      hartebeest    hippopotamus
##            1923            3217             327            2139             340
##     honeybadger    hyenaspotted    hyenastriped          impala          jackal
```

```
##             42           2752            112            691            304
##     koribustard        leopard      lionfemale       lionmale       mongoose
##            591             38           1519            665            290
##    monkeyvervet        ostrich       otherbird      porcupine       reedbuck
##              64            211           2322            114            946
##        reptiles      rhinoceros        rodents   secretarybird         serval
##             357              6            129            198            174
##            topi        warthog       waterbuck        wildcat      wildebeest
##             428           2685             25             48            689
##           zebra        zorilla
##            5592             9
```

For this lab you are going to choose two animals from the camera trap dataset. It could be a pair of animals that are predator and prey or two potential competitors. Once you decide on the two animals you want to compare you will make some predictions about how you think they will differ in their activity patterns.

Now we will use another function to download the camera trap photos and save them locally to our computer. You can change the season by changing the values for season (remember the available seasons are 1-4, 6-8,10,11). You can change the focal animal by changing the 'AnimalID'; make sure that the spelling and case is exactly the same as in the table above. You can change the number of photos per season that you download (the default is 5). When 'create.dir' is true this will create a folder in your current working directory. For this example the folder is called 'CameraTrapPhotoszebra'. If create.dir='FALSE' all the photos will be downloaded directly to your working directory. To find the directory type 'getwd()' into your R console.

```
CombinedAnimalDF <- CameraTrapDataAccess(urlpath= 'https://lilablobssc.blob.core.windows.net/snap
                    season= list(1,2),AnimalID='zebra', NumPhotos= 5,create.dir=TRUE)
```

Here we isolate only the columns from the dataframe that we need.

```
CombinedAnimalDF <- CombinedAnimalDF[,c("category_id","season","location","filename")]
head(CombinedAnimalDF)
```

```
##       category_id season location                                filename
## 1975        zebra     S1      C03 CameraTrapPhotoszebra/zebra_S1_C03.JPG
## 4742        zebra     S1      C06 CameraTrapPhotoszebra/zebra_S1_C06.JPG
## 292         zebra     S1      B05 CameraTrapPhotoszebra/zebra_S1_B05.JPG
## 7513        zebra     S1      D05 CameraTrapPhotoszebra/zebra_S1_D05.JPG
## 4582        zebra     S1      C06 CameraTrapPhotoszebra/zebra_S1_C06.JPG
## 45467       zebra     S2      E02 CameraTrapPhotoszebra/zebra_S2_E02.JPG
```

The function below will allow you to enter data and look through each photo included in the 'CombinedAnimalDF' spreadsheet and enter the time that the photo was taken. You can change option='Plot' to option='Viewer' to load the photos more quickly, but you will have to expand the photo to see the whole

thing. The input file should be the dataframe created using the 'CameraTrapDataAccess' function. You can break out of the function at any time by typibg 'break'. It will print out which row of the dataframe you were on when you stopped the function. If you would like to resume where you left off you must change 'rowstart=1' to the row number indicated, and change 'dataframe.cont = FALSE' to 'dataframe.cont = TRUE'.

Here is the function to view photos and annotate data.

```
CombinedAnimalDF_TimeAdded <- CameraTrapDataCollection(inputfile = CombinedAnimalDF,
                                                       rowstart = 1, dataframe.cont = I
```

Now you may want to save your datasheet locally

```
write.csv(CombinedAnimalDF_TimeAdded,'CombinedAnimalDF_TimeAdded.csv',row.names = F)
```

6.2 Part 2: Analyze your Serengeti camera trap data

First we load your data sheet with the times added. NOTE: Make sure that your updated datasheet has the exact same name as the file indicated below.

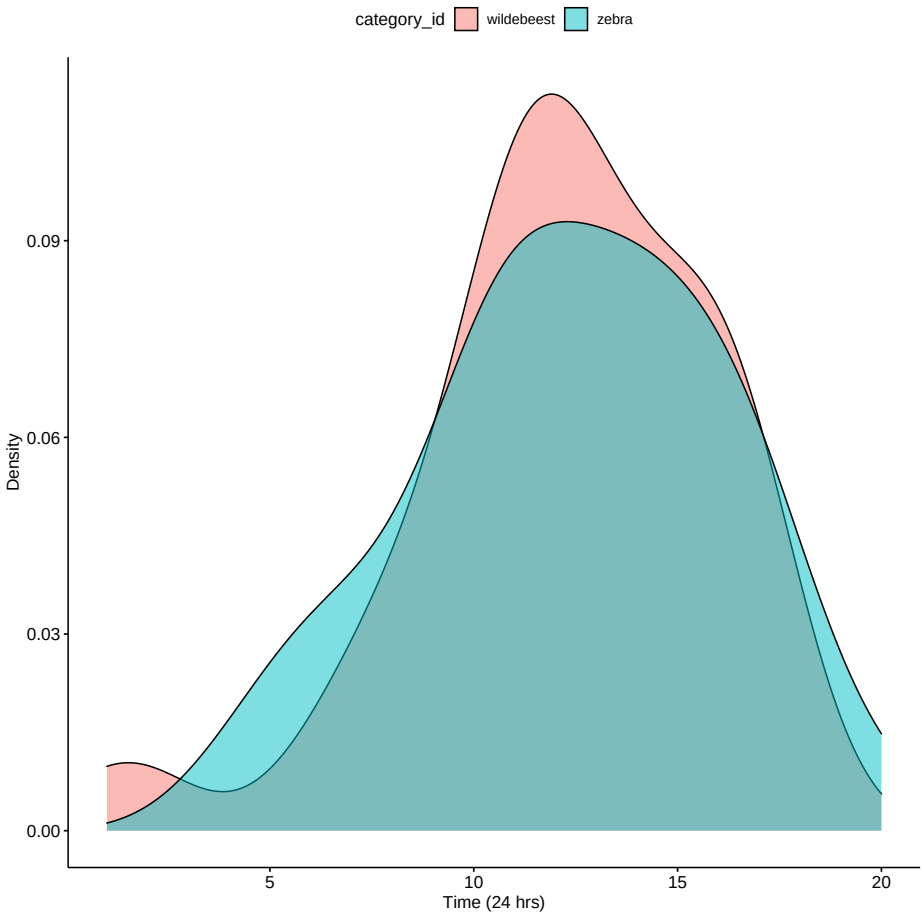
```
# You can read in your datasheet that you saved
CombinedDFTimes <- read.csv('CombinedAnimalDF_TimeAdded.csv')
```

Then we can check the structure

```
head(CombinedDFTimes)
```

Now we can make a density plot that will show the distribution of camera trap photos that were taken over 24-hours. We add the fill = 'category_id' so that we show different distributions for each animal.

```
ggdensity(data=CombinedDFTimes,x='Time',fill = 'category_id')+
  xlab('Time (24 hrs)') +ylab('Density')
```



Question 1. What do you notice about the overlap of the two density curves? Does it look like there is temporal niche partitioning?

Now we can calculate an overlap coefficient which can be used to investigate potential competitive and interaction possibilities between species.The value ranges from ranges from 0 (no overlap) to 1 (complete overlap).

First we subset our data to focus on the first animal in our dataset

```
FirstAnimal <- subset(CombinedDFTimes,category_id=='zebra')
```

Then we subset our data to focus on the second animal in our dataset

```
SecondAnimal <- subset(CombinedDFTimes,category_id=='wildebeest')
```

Now we use the overlap function to calculate the overlap coefficient

```
bayestestR::overlap(FirstAnimal$Time,SecondAnimal$Time)
```

```
## # Overlap
```

```
##
## 0.91
```

Question 2. How do you interpret the overlap coefficient for your data?

6.3 Part 3: Focus on your partner's Serengeti camera trap data

Now we will read in our partners data.
NOTE: Make sure that your updated datasheet has the exact same name as the file indicated below.

```
CombinedPartnerDFTimes <- read.csv('CombinedAnimalDF_TimeAddedPartner.csv')
```

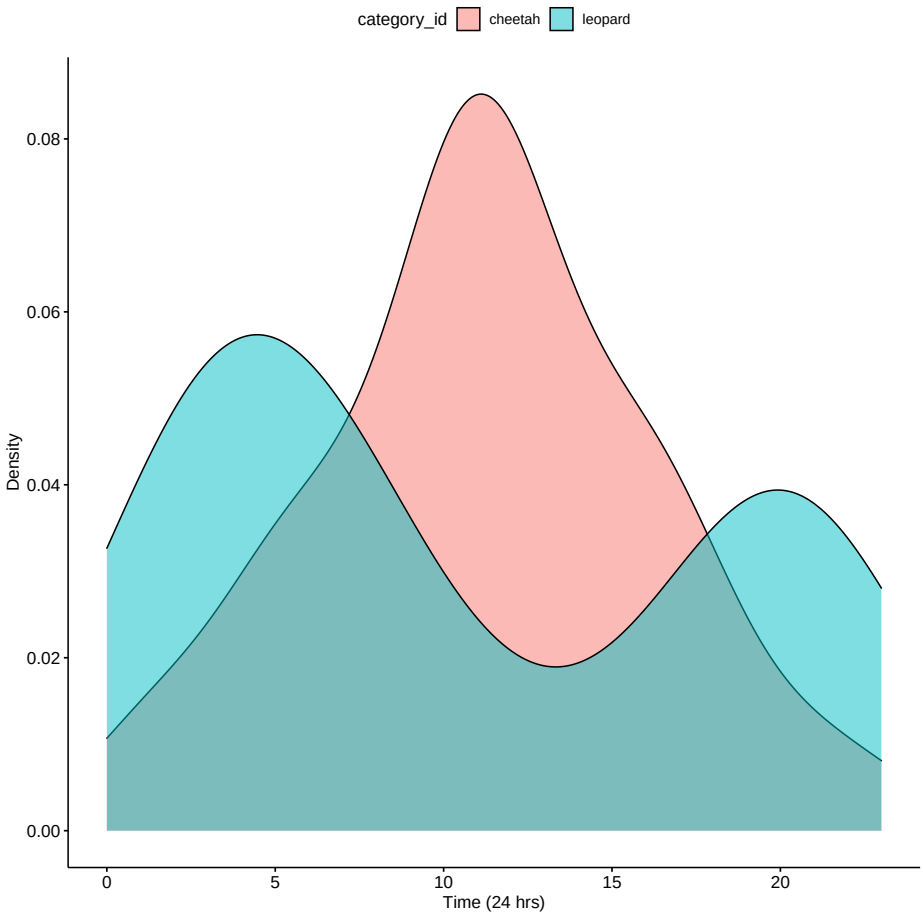
Let's check the structure

```
head(CombinedPartnerDFTimes)
```

```
##        X category_id season                            image_id location Time
## 1 247918     cheetah     S2   S2/R07/R07_R3/S2_R07_R3_IMAG0167      R07   15
## 2 326897     cheetah     S3 S3/K11/K11_R11/S3_K11_R11_IMAG0155      K11    1
## 3 204936     cheetah     S2   S2/J11/J11_R2/S2_J11_R2_IMAG1279      J11    6
## 4 364272     cheetah     S3 S3/P10/P10_R11/S3_P10_R11_IMAG0031      P10   12
## 5 262070     cheetah     S2   S2/U13/U13_R1/S2_U13_R1_PICT0091      U13    5
## 6 181602     cheetah     S2   S2/H07/H07_R3/S2_H07_R3_PICT4675      H07   16
```

Now we can make a density plot for our partner's data.

```
ggdensity(data=CombinedPartnerDFTimes,x='Time',fill = 'category_id')+
  xlab('Time (24 hrs)') +ylab('Density')
```



Question 3. What do you notice about the overlap of the two density curves for your partner's data? Does it look like there is temporal niche partitioning?

Now we can calculate an overlap coefficient which can be used to investigate potential competitive and interaction possibilities between species.The value ranges from ranges from 0 (no overlap) to 1 (complete overlap).

First we subset our data to focus on the first animal in your partner's data set

```
FirstAnimalPartner <- subset(CombinedPartnerDFTimes,category_id=='cheetah')
```

Then we subset our data to focus on the second animal in our dataset

```
SecondAnimalPartner <- subset(CombinedPartnerDFTimes,category_id=='leopard')
```

Now we use the overlap function to calculate the overlap coefficient

```
bayestestR::overlap(FirstAnimalPartner$Time,SecondAnimalPartner$Time)
```

```
## # Overlap
```

```
##
## 0.56
```

Question 4. What is the overlap coefficient for your partner's data? How do you interpret this?

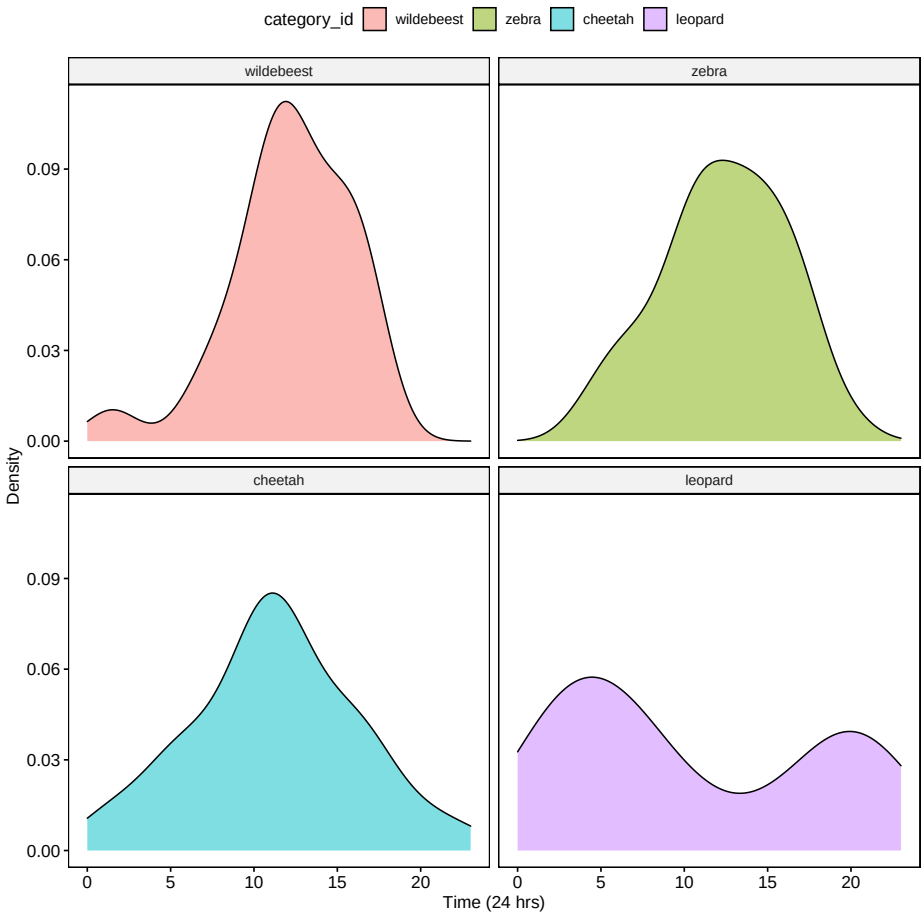
6.4 Part 4. Investigating temporal niche partitioning in four different animals

First we combine both datasets

```
AllDataCombined <- rbind.data.frame(CombinedDFTimes,CombinedPartnerDFTimes)
```

Then we plot the diel activity patterns for all four species

```
ggdensity(data=AllDataCombined,x='Time',fill = 'category_id',facet.by = 'category_id')+
  xlab('Time (24 hrs)') +ylab('Density')
```



Question 5. Based on the activity patterns and your understanding of the Serengeti food web how do you interpret these results? Is there evidence of temporal niche partitioning among potential competitors? What about interactions between potential predators and prey?

Appendix 1. Data exploration and visualization R script

```
# Please use the Lab 1 tutorial located here: https://bookdown.org/djc426/IntroToBehaviorTutorial

# Hello and welcome to your first R session!
# Remember any lines that include a # symbol will not be read by R,
# but provides information to the user (you!)

# First we load the relevant package
library(behaviouR)

## Lab 1a. Categorical data
# Here we create a simulated population with four categories (Infant, Juvenile, AdultFemale and A
DeerPopulationDF <- data.frame(DevelopmentStage=c('Infant','Juvenile','AdultFemale','AdultMale'),
                               NumberOfIndividuals=c(15,50,125,200))

# We then print the object so that we can see the output
DeerPopulationDF

# Now we want to plot the data. We will use a simple barplot to start.
ggbarplot(DeerPopulationDF, x='DevelopmentStage', y='NumberOfIndividuals')


## Lab 1b. Categorical and continuous data
# Load the dataset so that we can use it
data('MaleDeerRoarDF')

# We can check the structure of the dataframe by using the command 'head'
head(MaleDeerRoarDF)
```

```
# Now we will plot the categorical data with standard deviations.
ggbarplot(MaleDeerRoarDF, x='MaleCategory', y='RoarsPerMinute',
          add = c("mean_sd"),xtickslab.rt = 90)

# Now we will plot the categorical data with colors for each category.
ggbarplot(MaleDeerRoarDF, x='MaleCategory', y='RoarsPerMinute', fill = 'MaleCategory',
          add = c("mean_sd"),xtickslab.rt = 90)

# Now we will plot the categorical data with user-specified colors for each category.
ggbarplot(MaleDeerRoarDF, x='MaleCategory', y='RoarsPerMinute', fill = 'MaleCategory',
          palette = c('red','gray','white','black'),
          add = c("mean_se"),xtickslab.rt = 90)


## Lab 1c. Categorical and continuous data
# We will simulate a dataset for males of different weight and harem size.

# The function below simulates our data. N is the number of individuals,
# CorrelationCoefficient tells us how correlated our data are,
# MaleMeanBodyWeight is the mean body weight of males in our population and
# MaleReproductiveSuccess is the mean number of females in the harem.
MaleRedDeerDF <- CorrelatedDataSimulationFunction(N=100,
                                                  CorrelationCoefficient= 0.45,
                                                  MaleMeanBodyWeight = 125,
                                                  MaleReproductiveSuccess = 3)

# We can check the output
head(MaleRedDeerDF)

# Make a scatterplot of the data.
ggscatter(data=MaleRedDeerDF,x='MaleBodyWeight',y='MaleReproductiveSuccess')

# Make a scatterplot with a trendline.
ggscatter(data=MaleRedDeerDF,x='MaleBodyWeight',y='MaleReproductiveSuccess',
          add='reg.line')

# Create a linear model where MaleBodyWeight is the independent variable
# and ReproductiveSuccess is the dependent variable.
MaleDeerModel <- lm(MaleReproductiveSuccess ~ MaleBodyWeight,data=MaleRedDeerDF)

# We can look at the output of the model
MaleDeerModel

# Create a null model and a model with MaleBodyWeight as a predictor.
MaleDeerNull <- lm(MaleReproductiveSuccess ~ 1, data=MaleRedDeerDF)
```

```
MaleDeerModel <- lm(MaleReproductiveSuccess ~ MaleBodyWeight,data=MaleRedDeerDF)

# Compare models using AIC.
AICctab(MaleDeerModel,MaleDeerNull, weights=T)

## Lab 1d. Multivariate data
# Just as before we load our data
data("DeerSpeciesAcousticFeatures")

# Check the structure
head(DeerSpeciesAcousticFeatures)

# Here is our modified dataset that we will use for PCA
DeerSpeciesAcousticFeatures[,-c(4)]

# Run the PCA using the 'princomp' function
DeerSpeciesAcousticFeaturesPCA <- princomp(DeerSpeciesAcousticFeatures[,-c(4)])

# Plot the results of our PCA.
autoplot(DeerSpeciesAcousticFeaturesPCA, data = DeerSpeciesAcousticFeatures, colour = 'Class',
         loadings = FALSE)+theme_bw()

# Plot the PCA with arrows indicating which features are important for distinguishing between gro
autoplot(DeerSpeciesAcousticFeaturesPCA, data = DeerSpeciesAcousticFeatures, colour = 'Class',
         loadings = TRUE, loadings.colour = 'red',
         loadings.label = TRUE,
         loadings.label.size = 5)+theme_bw()
```