Thesis Script

Maria Laura Mahecha Escobar

Table of contents

0.1 Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

0.2 Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

For running this script you need the following packages installed in your workspace:

```r
library(knitr)
library(circular)
```

```
Warning: package 'circular' was built under R version 4.2.2
```

```
Attaching package: 'circular'
```

```
The following objects are masked from 'package:stats':

    sd, var
```

```r
library(ggplot2)
library(rmarkdown)
library(quarto)
```

```
Warning: package 'quarto' was built under R version 4.2.2
```

```r
library(lmerTest)
```

```
Warning: package 'lmerTest' was built under R version 4.2.2
```

```
Loading required package: lme4
```

```
Loading required package: Matrix
```

```
Attaching package: 'lmerTest'
```

```
The following object is masked from 'package:lme4':

    lmer
```

```
The following object is masked from 'package:stats':

    step
```

```
library(cowplot)
```

```
Warning: package 'cowplot' was built under R version 4.2.2
```

```
library(lme4)
library(dplyr)
```

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

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(janitor)
```

```
Warning: package 'janitor' was built under R version 4.2.2
```

```
Attaching package: 'janitor'
```

```
The following objects are masked from 'package:stats':

    chisq.test, fisher.test
```

```
library(tvthemes)
```

```
Warning: package 'tvthemes' was built under R version 4.2.2
```

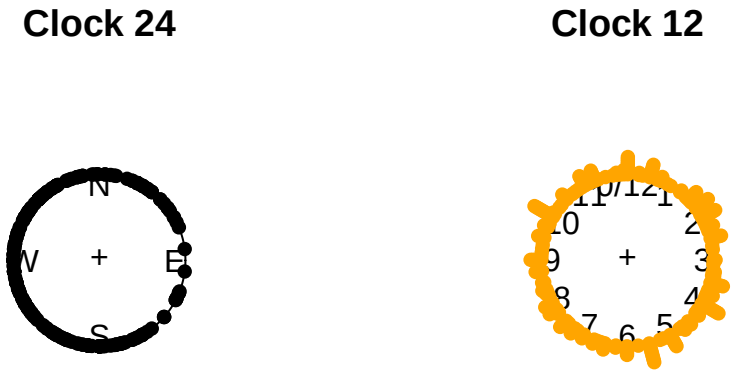
```
library(phytools)
```

```
Loading required package: ape
```

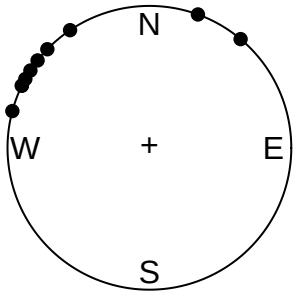
```
Loading required package: maps
```

The following code shows an example data for the package “circular” to draw a plot of @fisherB1c

```
data(fisherB1c)
par(mfcol=c(1,2))
plot(fisherB1c, main="Clock 24", shrink=1.5, template = "geographics")
plot(fisherB1c, template="clock12", main="Clock 12", shrink=1.5, stack=T, bins=720, pch=16
```

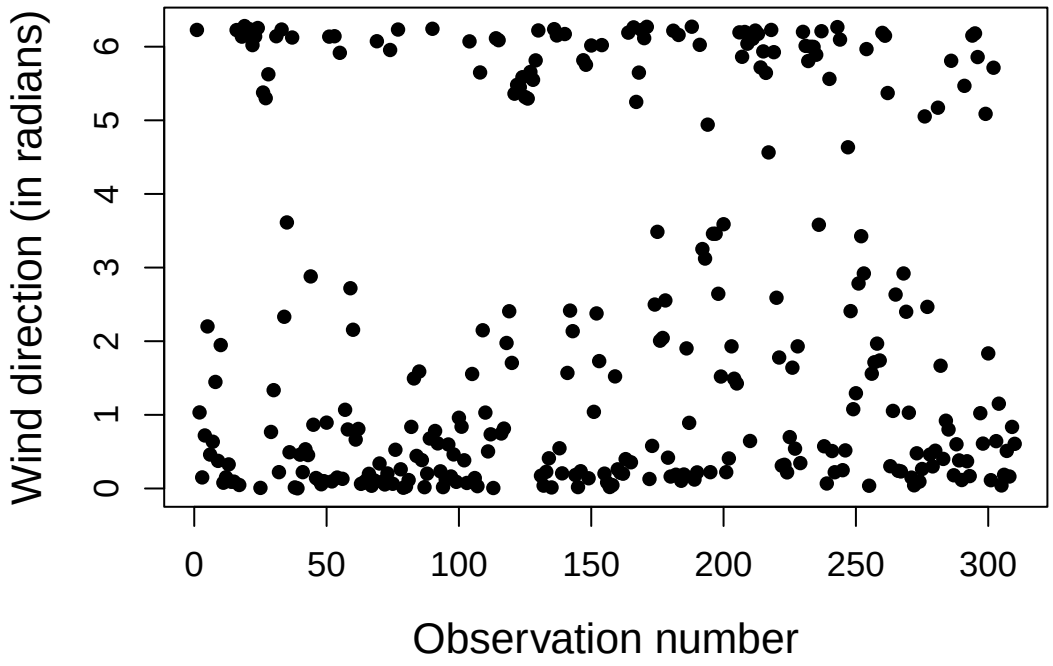



```
data(turtles)
turtles[,2] <- circular(turtles[,2], units='degrees', template='geographics')
plot(turtles[,2])
```




```
windc<-circular(wind, type="angles", units="radians", template='geographics')
par(mai=c(0.85, 0.85, 0.05, 0.05), cex.axis=1.1, cex.lab=1.3)

plot(wind, pch=16, xlab="Observation number", ylab="Wind direction (in radians)")
```

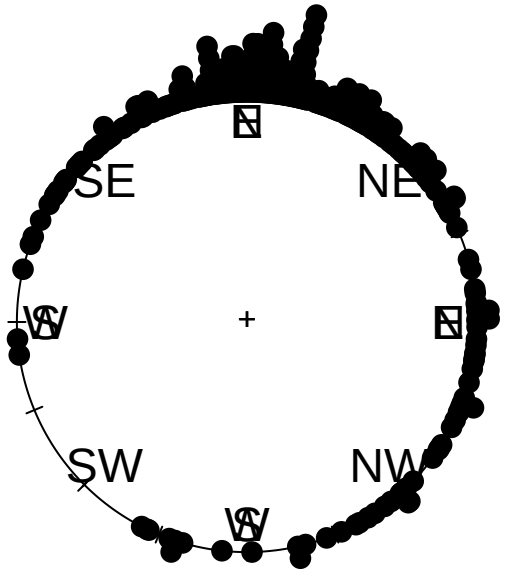



```
par(mai=c(0, 0, 0, 0))

plot(windc, cex=1.5, bin=720, stack=TRUE, sep=0.035, shrink=1.3)

axis.circular(at=circular(seq(0, 7*pi/4,pi/4)), labels=c("N","NE","E","SE","S","SW","W","N

ticks.circular(circular(seq(0,2*pi,pi/8)), zero=pi/2, rotation='clock', tcl=0.075)
```



For exploring data plot this:

```
#ggplot(data=TestData_LMEM, aes(x=Dia, y=Intervalos.activos, color=Control.o.no))+
  #geom_point() + geom_line()
```

For Lineal mixed effects models we can use this code (HOW TO INTERPRET THEM!!!???))

```
#TestData_LMEM<-(LMEM_NotRealActivityData)
#TestModelLMEL_1<-lmer(Intervalos.activos ~ Control.o.no+DiaNoche+(1|Ring), REML=F, data=T
#summary(TestModelLMEL_1)
#TestModelLMEL_2<-lmer(Intervalos.activos ~ Control.o.no*DiaNoche+(1|Ring), REML=F, data=T
#summary(TestModelLMEL_2)
#anova(TestModelLMEL_1,TestModelLMEL_2)
```

For ploting an actogram

```
# cargar arbol
#arbol <- read.tree(file.choose())
#arbol
#class(arbol)
#arbol.1 <- arbol[[1]]
#arbol.1
```

```
#write.tree(new.arbol.GEN)

#arbol.1<- read.nexus("C:\\MLME\\Andes\\TesisBiol\\todo_24_mayo_19\\arbolGEN.tre")
#arbol.1<- read.nexus("D:\\LaBodega\\DeMariaLaura\\Andes\\tesis\\biologia\\arbolGEN.tre")

#data.mod.GEN <- mean.data[which(!is.na(mean.data$Diet.Inv)),]
#head(data.mod.GEN)

#species.GEN <- unique(data.mod.GEN$Jetz_name)

#sp.to.rm.GEN <- arbol.1$tip.label[!(arbol.1$tip.label %in% species.GEN)] #To remove speci

#new.arbol.GEN <- drop.tip(arbol.1,tip=sp.to.rm.GEN)
#plot(new.arbol.GEN)
#write.tree(new.arbol.GEN, "C:\\Users\\J303\\Documents\\LauraMahecha\\arbolGEN.tre")
#write.tree(new.arbol.GEN, "C:\\MLME\\Andes\\TesisBiol\\todo_24_mayo_19\\arbolGEN2.tre")

#data.mod.GEN.F <- mean.data[which(!is.na(sp_dietFruit$Diet.Fruit)),]
#head(data.mod.GEN.F)

#species.GEN.F <- unique(data.mod.GEN.F$Jetz_name)

#sp.to.rm.GEN.F <- arbol.1$tip.label[!(arbol.1$tip.label %in% species.GEN.F)]

#new.arbol.GEN.F <- drop.tip(arbol.1,tip=sp.to.rm.GEN.F)
#new.arbol.GEN.F
#write.tree(new.arbol.GEN.F, "C:\\Users\\J303\\Documents\\LauraMahecha\\arbolGEN.tre"))
```

0.3 Emlen funnel analysis

Trying to taking the emlen excel to R:

To calculate the mean direction, r value and p value for each emlen funnel experiments we need to know if the distribution is unimoda or axial (bimodal). You can notice this in the paper when you count scratches or using mathematical functions. To know if you have an axial distribution your angles need to be doubled and then compare the values from them with an unimodal distribution.

Final vectors that contain the values for the sin and cos for angle 1 and 2 in the sector and that we are going to use from this part of the code are:

@sen24sector_1
@cos24sector_1
@sen24sector_2
@cos24sector_2

```
rad2deg <- function(rad) {(rad * 180) / (pi)} #Converts radians to degrees. Font:https://s
deg2rad <- function(deg) {(deg * pi) / (180)} #converts degrees to radians. Font:https://s

####Sin and cos 24 sectors angles####
sectorAngle_1_deg<- as.vector(c(0,15,30,45,60,75,90,105,120,135,150,165,180,195,210,225,24
sectorAngle_1<-deg2rad(sectorAngle_1_deg) #Converts angle degrees into radians to calculat

sectorAngle_2_deg<- as.vector(c(0,30,60,90,120,150,180,210,240,270,300,330,360,390,420,450
sectorAngle_2<-deg2rad(sectorAngle_2_deg) #Converts angle degrees into radians to calculat


###Sin sector angle 1###
sen24sector_1<-as.vector(sin(sectorAngle_1)) #It contains the values of the sin for each a
print.default(sen24sector_1, digits = 3) #Prints the result with only 3 digits
```

```
 [1]  0.00e+00  2.59e-01  5.00e-01  7.07e-01  8.66e-01  9.66e-01  1.00e+00
 [8]  9.66e-01  8.66e-01  7.07e-01  5.00e-01  2.59e-01  1.22e-16 -2.59e-01
[15] -5.00e-01 -7.07e-01 -8.66e-01 -9.66e-01 -1.00e+00 -9.66e-01 -8.66e-01
[22] -7.07e-01 -5.00e-01 -2.59e-01
```

```
###cos sector angle 1###
cos24sector_1<-as.vector(cos(sectorAngle_1)) #It contains the values of the cos for each a
print.default(cos24sector_1, digits = 3) #Prints the result with only 3 digits
```

```
 [1]  1.00e+00  9.66e-01  8.66e-01  7.07e-01  5.00e-01  2.59e-01  6.12e-17
 [8] -2.59e-01 -5.00e-01 -7.07e-01 -8.66e-01 -9.66e-01 -1.00e+00 -9.66e-01
[15] -8.66e-01 -7.07e-01 -5.00e-01 -2.59e-01 -1.84e-16  2.59e-01  5.00e-01
[22]  7.07e-01  8.66e-01  9.66e-01
```

```
###sin sector angle 2###
sen24sector_2<-as.vector(sin(sectorAngle_2)) #It contains the values of the sin for each a
print.default(sen24sector_2, digits = 3) #Prints the result with only 3 digits
```

```
 [1]  0.00e+00  5.00e-01  8.66e-01  1.00e+00  8.66e-01  5.00e-01  1.22e-16
```

```
 [8] -5.00e-01 -8.66e-01 -1.00e+00 -8.66e-01 -5.00e-01 -2.45e-16  5.00e-01
[15]  8.66e-01  1.00e+00  8.66e-01  5.00e-01  3.67e-16 -5.00e-01 -8.66e-01
[22] -1.00e+00 -8.66e-01 -5.00e-01
```

```
###cos sector angle 2###
cos24sector_2<-as.vector(cos(sectorAngle_2)) #It contains the values of the cos for each a
print.default(cos24sector_2, digits = 3) #Prints the result with only 3 digits
```

```
 [1]  1.00e+00  8.66e-01  5.00e-01  6.12e-17 -5.00e-01 -8.66e-01 -1.00e+00
 [8] -8.66e-01 -5.00e-01 -1.84e-16  5.00e-01  8.66e-01  1.00e+00  8.66e-01
[15]  5.00e-01  3.06e-16 -5.00e-01 -8.66e-01 -1.00e+00 -8.66e-01 -5.00e-01
[22] -4.29e-16  5.00e-01  8.66e-01
```

In this section we are going to calculate basic trigonometric functions for posterior analysis.
From this part we are interested in:

@arctan_angle1
@arctan_angle2
x an y values for both angles

```
EmlenFunnels_RawData<-read.table("E:\\Users\\mlmah\\OneDrive\\Documentos\\MLME\\Maestria\\
EmlenFunnel<- select(EmlenFunnels_RawData, c("Ring","S1","S2","S3","S4","S5","S6","S7","S8

sectors<-as.data.frame(select(EmlenFunnels_RawData, c("S1","S2","S3","S4","S5","S6","S7","

##angle 1##
senSectors_angle1<-mapply("*",as.data.frame(sectors),sen24sector_1) #Multiplies each colum
senData_angle1<-rowSums(senSectors_angle1) #It sums each column and creates a vector with
y_angle1<-senData_angle1/EmlenFunnel$TotalScratches #Calculates the y value for angle 1

cosSectors_angle1<-mapply("*",as.data.frame(sectors),cos24sector_1) #Multiplies each colum
cosData_angle1<-rowSums(cosSectors_angle1) #It sums each column and creates a vector with
x_angle1<-cosData_angle1/EmlenFunnel$TotalScratches #Calculates the x value for angle 1

tan_angle1<-(y_angle1/x_angle1) #Calculates the tangent for angle 1
arctan_angle1<-atan2(y_angle1,x_angle1)*(180/pi) #Calculates the arctan for angle 1
pre_dir_1<-ifelse(arctan_angle1<0, arctan_angle1+360, arctan_angle1) #Calculates the direc

##angle2##
senSectors_angle2<-mapply("*",as.data.frame(sectors),sen24sector_2) #Multiplies each colum
senData_angle2<-rowSums(senSectors_angle2) #It sums each column and creates a vector with
```

```
y_angle2<-senData_angle2/EmlenFunnel$TotalScratches #Calculates the y value for angle 2

cosSectors_angle2<-mapply("*",as.data.frame(sectors),cos24sector_2) #Multiplies each colum
cosData_angle2<-rowSums(cosSectors_angle2) #It sums each column and creates a vector with
x_angle2<-cosData_angle2/EmlenFunnel$TotalScratches #Calculates the x value for angle 2

tan_angle2<-(y_angle2/x_angle2) #Calculates the tangent for angle 2
arctan_angle2<-atan2(y_angle2,x_angle2)*(180/pi) #Calculates the arctan for angle 2
pre_dir_2<-ifelse(arctan_angle2<0, arctan_angle2+360, arctan_angle2) #Calculates the direc
pre_dir_2a<-pre_dir_2/2 #Calculates the first direction of a bimodal distribution
pre_dir_2b<-pre_dir_2a+180 #Calculates the second direction of a bimodal distribution
```

In this section we are going to handling with axial data. Axial data returns two different values for direction and we need to choose only one for it. In this case, the way of doing so is selecting the angle that has most of the scratches. You can do this manually and modifying your database or you can use the code below. This code searches for the values in the original database and compares which one is higher. Then writes the obtained direction from it.

```
#To know the nearest value and find the direction in axial data

###2a###
# Create a vector to store the closest positions for direction 2a
closest_positions_2a <- numeric(length(pre_dir_2a))
for (i in 1:length(pre_dir_2a)) {# Calculate the distance to each value in sectorAngel_1_d
  distances_2a <- abs(pre_dir_2a[i] - sectorAngle_1_deg)# Find the position of the closest
  closest_positions_2a[i] <- which.min(distances_2a)
}
print(closest_positions_2a) #prints the position of the closest value for the angle given
```

```
[1] 11  1  3  6
```

```
closest_positions_2a <-as.vector(paste("S", closest_positions_2a, sep="")) #Put an "S" bef

###2b###
# Create a vector to store the closest positions for direction 2b
closest_positions_2b <- numeric(length(pre_dir_2b))
for (i in 1:length(pre_dir_2b)) {# Calculate the distance to each value in sectorAngel_1_d
  distances_2b <- abs(pre_dir_2b[i] - sectorAngle_1_deg)# Find the position of the closest
  closest_positions_2b[i] <- which.min(distances_2b)
}
print(closest_positions_2b) #prints the position of the closest value for the angle given
```

```
[1] 23 13 15 18
```

```
closest_positions_2b<-as.vector(paste("S", closest_positions_2b, sep="")) #Put an "S" befo


#To extract number of scratches and compare
  #For direction_2a
scratches_2a <- vector() #create an empty vector of unknown dimensions
for (i in seq_along(closest_positions_2a)) { #For a sequence of data in the vector
  scratches_2a <- c(scratches_2a, EmlenFunnel[i, closest_positions_2a[i]]) #match the name
}
scratches_2a
```

```
[1]  18 115  23  30
```

```
  #For direction_2b
scratches_2b <- vector() #create an empty vector of unknown dimensions
for (i in seq_along(closest_positions_2b)) { #For a sequence of data in the vector
  scratches_2b <- c(scratches_2b, EmlenFunnel[i, closest_positions_2b[i]]) #match the name
}
scratches_2b
```

```
[1] 25  3 15  5
```

```
bimodal_dir<-ifelse(scratches_2a<scratches_2b,pre_dir_2b, pre_dir_2a) #If the number of sc
```

In this section we will calculate the mean lenght of the vectors (r) and the overall direction of the individual (@directionEmlen)

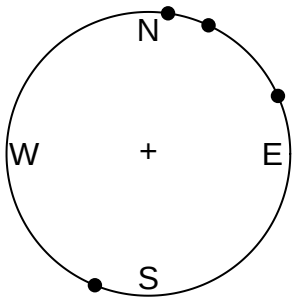
```
r_value_1<-sqrt(((x_angle1)^2)+((y_angle1)^2)) #Calculates the r value for angle 1
r_value_2<-sqrt(((x_angle2)^2)+((y_angle2)^2)) #calculates the r value for both angles in
rvalue<-ifelse(r_value_1>r_value_2, r_value_1, r_value_2)
rvalue
```

```
[1] 0.2735225 0.7656407 0.1375177 0.1090403
```

```
directionEmlen<- ifelse(r_value_1>r_value_2,pre_dir_1, bimodal_dir) #compares r values and
directionEmlen<- as.circular(directionEmlen, units='degrees', template='geographics')
```

```
Warning in as.circular(directionEmlen, units = "degrees", template = "geographics"): an obje
  type: 'angles'
  modulo: 'asis'
  zero: 0
  rotation: 'counter'
evalexprenvirenclos
```

```
plot.circular(directionEmlen, units='angles', template='geographics')
```




```
pvalue_1_uniformity<-((-EmlenFunnel$TotalScratches)*(r_value_1)^2)
pvalue_2_uniformity<-((-EmlenFunnel$TotalScratches)*(r_value_2)^2)
pvalue_uniformity<-ifelse(r_value_1>r_value_2, pvalue_1_uniformity,pvalue_2_uniformity)


EmlenFunnels_withDir<-cbind(EmlenFunnels_RawData,directionEmlen,rvalue,pvalue_uniformity)
EmlenFunnels_withDir
```

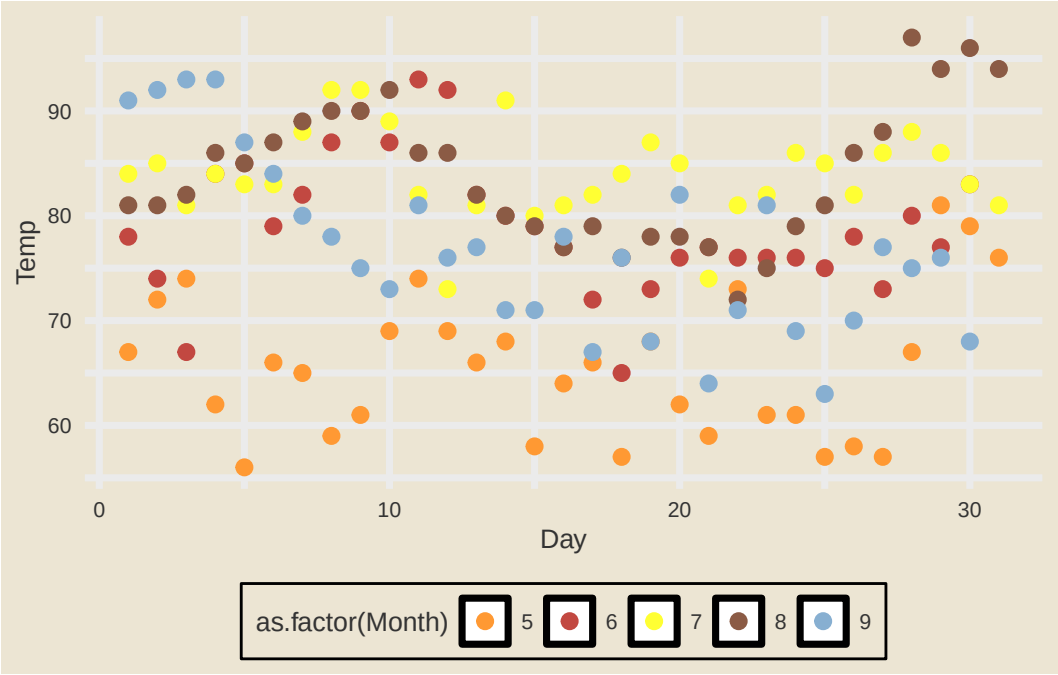
```
  CaptureDate  Species BreedingDistribution     Ring Treatment Recording Line
1  15/04/2023     Pied                polar 1EA76002   control         1    D
2  15/04/2023     Pied                polar 1EA76002  extended         1    E
```

```
3  15/04/2023 Collared              subpolar 1EA76003   control          1    D
4  15/04/2023 Collared              subpolar 1EA76003  extended          1    D
   S1  S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S20 S21
1   3   1  1  1  5  9  9 10 14  22  18  24  18  18  14  17  21  20  11   6   3
2 115 126 18 11 21 18 19  8  3   4   8   1   3   2   1   0   0   1   2   1  17
3  25  34 23 17 17 23 19  6 10   2   9  17  27  18  15  25  17  15   9  17  22
4   7   8 11 10 24 30 18 18 11   4   7  13  15  10  13  14  12   5   8   5  19
  S22 S23 S24 Max TotalScratches directionEmlen    rvalue pvalue_uniformity
1  13  25  19  NA            302     202.152136 0.2735225        -22.594000
2   9  28 102  NA            518       7.962591 0.7656407       -303.654566
3  33   9  13  NA            422      25.074688 0.1375177         -7.980495
4  20  18   7  NA            307      65.920781 0.1090403         -3.650166
```

```
###AHORA LOS PINCHES MODELOS MARIA LAURITA###
```

Color palette for the figures presented

```
ggplot(airquality, aes(x = Day, y = Temp,
     group = as.factor(Month), color = as.factor(Month))) +
     geom_point(size = 2.5) +
     scale_color_avatar(palette = "AirNomads") +
     theme_avatar()
```



0.4 Cuartito de San Alejo

This is the part of the code where I put everything I tried and does not worked but I don't want to delete

```
#sen24sector_1 <-as.vector(sin(c(0,15,30,45,60,75,90,105,120,135,150,165,180,195,210,225,2
#print.default(sen24sector_1, digits = 3)
#deg2rad(sen24sector_1)

#sen24sector_1 <- lapply(sen24sector_1, "*", (3.14/180))
#cos24sector_1=   #It has the values for the cos for each angle in the 24 sectors

#cos_angle1= cos*  #This line multiplies each cos for the angle 1 with the number of scrat
#y1=
#arcotan_2<-((atan2(y1,x1))*(180/pi)) #This code takes out the arcotangent from two axis:

#______


#{
 # axial_dir_a<-(arctan_angle2+360)/2
  #  axial_dir_b<-(arctan_angle2)/2
   # pvalue_2<-((-EmlenFunnel$TotalScratches)*(r_value_2)^2)
#} else if (arctan_angle1<0){
 #   unimodal_dir<-arctan_angle1+360
  #  pvalue_1<-((-EmlenFunnel$TotalScratches)*(r_value_1)^2)
  #} else {
   # unimodal_dir<-arctan_angle1
    #pvalue_1<-((-EmlenFunnel$TotalScratches)*(r_value_1)^2)}


#______



#if(r_value_1<r_value_2) {
 # axial_dir_a<-(arctan_angle2+360)/2
  #  axial_dir_b<-(arctan_angle2)/2
   # pvalue_2<-((-EmlenFunnel$TotalScratches)*(r_value_2)^2)
#} else if (arctan_angle1<0){
 #   unimodal_dir<-arctan_angle1+360
  #  pvalue_1<-((-EmlenFunnel$TotalScratches)*(r_value_1)^2)
```

```
  #} else {
   # unimodal_dir<-arctan_angle1
  #  pvalue_1<-((-EmlenFunnel$TotalScratches)*(r_value_1)^2)}

#______


#axial_dir_a<-ifelse(r_value_1<r_value_2, (arctan_angle2+360)/2, NA)
#axial_dir_a
#axial_dir_b

#ifelse(r_value_1<r_value_2, {axial_dir_a<-(arctan_angle2+360)/2; axial_dir_b<-(arctan_ang
 #      ifelse(arctan_angle1<0, {unimodal_dir<-arctan_angle1+360; pvalue_1<-((-EmlenFunnel

#axial_dir_a
#axial_dir_b
#pvalue_2
#unimodal_dir
#pvalue_1

#______


# Create a dataframe with some example data
df <- data.frame(values = c(2, 5, 8, 11, 14))

# Create a vector of positions to retrieve
positions <- c(1, 3, 5)

# Retrieve the values from the dataframe based on the positions
retrieved_values <- df$values[positions]

# Print the retrieved values
print(retrieved_values)
```

```
[1]  2  8 14
```

```
#________


# Create two vectors
```

```r
vector1 <- c(2, 5, 8, 11, 12)
vector2 <- c(3, 6, 9, 12, 15)

# Create a vector to store the closest positions
closest_positions <- numeric(length(vector1))

# Loop over each value in vector1
for (i in 1:length(vector1)) {
  # Calculate the distance to each value in vector2
  distances <- abs(vector1[i] - vector2)
  # Find the position of the closest value
  closest_positions[i] <- which.min(distances)
}

# Print the resulting vector of closest positions
print(closest_positions)
```

```
[1] 1 2 3 4 4
```

```r
#___________



# Create a dataframe with some example data
df <- data.frame(x = c(1, 2, 3),
                 y = c(4, 5, 6),
                 z = c(7, 8, 9))

# Create a vector of column names to match, where each position in the vector represents a
col_names <- c("z", "x", "y")

# Initialize an empty vector to store the extracted values
extracted_values <- vector()

# Use a for loop to extract values for each column name in the vector
for (i in seq_along(col_names)) {
  extracted_values <- c(extracted_values, df[i, col_names[i]])
}

# Print the extracted values
print(extracted_values)
```

```
[1] 7 2 6
```

```
#_______
```