S.5 part 1. Code for generating quantitative predictions with each model

```
#Libraries Used
library(ggplot2)
library(car)
library(NLRoot)
#install.packages("NLRoot")
library(rootSolve)
library(psych)

###########################Generating Quantitative Predictions with each Model
#Reading in DeVault et al. 2015 empirical data used to make quantitative predictions
devault_data<-read.csv("DeVaultData.csv", header=TRUE)

#Renaming columns
colnames(devault_data)<-c("Fate", "AD","FID","Size","Speed", "HQA","TTCA","TTCF")

#Perceptual Limits Hypothesis
#r is the radius of a vehicle width
r<-1.725/2
#a is the inverse of a species spatial resolving power
a<-1/4.82
#dd is the equation to estimate detection distance based on a species visual acuity
dd<-(r)/(tan(a/2)*(pi/180))
dd

########################################FEAR Hypothesis

#Function that calculates phi index values and p-value for the phi-index for the FEAR hypothesis.
#code provided by (Samia et al. 2014)

phi.index<-function(data, rounds){

 data = subset(data, data$FID & data$AD !='Na')
 N = nrow(data)
 phi = numeric(0)                    #creates temporary vector to store simulated phi-indices
 progress.bar <- txtProgressBar(min = 0,
                         max = rounds,
                         style = 3)     #insert progress bar

 #null expectation
 #--------------------------------------------------
 for(i in 1:rounds){

  setTxtProgressBar(progress.bar, i)          #start progress bar

  sFID = runif(nrow(data),
          0,
          data$AD)                  #simulates random FIDs

  phi[i] = 1 - (sum((data$AD-sFID)/data$AD)/ N)
```

```
 } #close i

 close(progress.bar)                          #close progress bar


 #computes P-value
 #---------------------------------------------------
 obs.phi = 1 - (sum((data$AD-data$FID)/data$AD)/ N)   #extract observed phi
 P  = sum(phi >= obs.phi)/rounds                #calculates the P-value of observed phi


 #plot
 #---------------------------------------------------
 plot(data$AD,
      data$FID,
      xlab = "AD",
      ylab = "FID",
      las = 1,
      bty = "l",
      ylim=c(0,max(data$AD)),
      xlim=c(0,max(data$AD)),
      t="n")
 abline(0, 1, col=8, lwd=3, lty = "dotted")
 points(data$AD, data$FID, cex = 1.3, pch = 21, bg=16)


 #output to workspace
 #---------------------------------------------------
 output = list('phi index '= obs.phi,
              'P-value'=P,
              'sample size' = N)

 return(output)


} # close function


#####Estimating phi and its p-value per each AD~FID per a given vehicle approach speed treatment with

#Number of simulated p-values for the phi index to compare the empirically observed phi against
rounds<-1000

#Data per each speed treatment

#60 km/h speed treatment
data_60<-devault_data[1:20,]

#90 km/h speed treatment
data_90<-devault_data[21:40,]
```

```
#120 km/h speed treatment
data_120<-devault_data[41:60,]

#150 km/h speed treatment
data_150<-devault_data[61:80,]

#180 km/h speed treatment
data_180<-devault_data[81:100,]

#210 km/h speed treatment
data_210<-devault_data[101:120,]

#240 km/h speed treatment
data_240<-devault_data[121:130,]

#360 km/h speed treatment
data_360<-devault_data[131:140,]


#Linear Regression between AD & FID for each speed treatment with the intercept forced to be 0

#60 km/h speed treatment
model.60<-lm(FID~AD+0, data=data_60)
summary(model.60)

#90 km/h speed treatment
model.90<-lm(FID~AD+0, data=data_90)
summary(model.90)

#120 km/h speed treatment
model.120<-lm(FID~AD+0, data=data_120)
summary(model.120)

#150 km/h speed treatment
model.150<-lm(FID~AD+0, data=data_150)
summary(model.150)

#180 km/h speed treatment
model.180<-lm(FID~AD+0, data=data_180)
summary(model.180)

#210 km/h speed treatment
model.210<-lm(FID~AD+0, data=data_210)
summary(model.210)

#240 km/h speed treatment
model.240<-lm(FID~AD+0, data=data_240)
summary(model.240)

#360 km/h speed treatment
model.360<-lm(FID~AD+0, data=data_360)
```

```
summary(model.360)


#Calculating the phi index value and significance for each speed treatment of DeVault et al. 2015

#60 km/h speed treatment
output.60<-phi.index(data_60,rounds)

#90 km/h speed treatment
output.90<-phi.index(data_90,rounds)

#120 km/h speed treatment
output.120<-phi.index(data_120,rounds)

#150 km/h speed treatment
output.150<-phi.index(data_150,rounds)

#180 km/h speed treatment
output.180<-phi.index(data_180,rounds)

#210 km/h speed treatment
output.210<-phi.index(data_210,rounds)

#240 km/h speed treatment
output.240<-phi.index(data_240,rounds)

#360 km/h speed treatment
output.360<-phi.index(data_360,rounds)


#Assempling the output for the phi index based on DeVault et al. 2015 data in a single data frame

output<-
data.frame(rbind(output.60,output.90,output.120,output.150,output.180,output.210,output.240,output.360
))
output$Speed<-c(60,90,120,150,180,210,240,360)
output$phi.index.<-as.numeric(output$phi.index.)
output$P.value<-as.numeric(output$P.value)
output$sample.size<-as.numeric(output$sample.size)
output$coef<-as.numeric(c(model.60[1], model.90[1],model.120[1], model.150[1],model.180[1],
model.210[1], model.240[1],model.360[1]))

output$phi.index./output$coef


#Plot Phi-Index values for each speed treatment

phi.graph<-ggplot(data=output, aes(x = Speed, y=phi.index., label= round(phi.index.,2)))+
  geom_text(hjust = -0.25, vjust = 0.75, nudge_x = -0.05, size=4.5)+
  geom_smooth(method="lm", formula= y~x)+ geom_point(size=2)+
  labs(y = "Φ Index",x = "Vehicel Speed (km/h)")+
```

```
  theme_classic(base_size = 16)+
  scale_x_continuous(breaks=c(0, 50, 100, 150, 200, 250, 300, 350))+
  ylim(0,1)+geom_hline(yintercept = 0.5,linetype="dashed",size=1)
phi.graph

#AD effect on FID according to a linear model for each speed treatment
coef.graph<-ggplot(data=output, aes(x = Speed, y=coef, label= round(coef,2)))+
  geom_text(hjust = -0.25, vjust = 0.75, nudge_x = -0.05, size=4)+
  geom_smooth(method="lm", formula= y~x)+ geom_point(size=2)+
  labs(y = "coef",x = "Vehicel Speed (km/h)")+
  theme_classic(base_size = 16)+
  scale_x_continuous(breaks=c(0, 50, 100, 150, 200, 250, 300, 350))+
  ylim(0,1)+geom_hline(yintercept = 0.5,linetype="dashed",size=1)
coef.graph



###Estimated FID based on the AD where 95% of Prey(Cowbirds) became alert


#Identifying the distance at which 95% of prey had become alert

#60 km/h speed treatment
AD.60<-quantile(devault_data$AD[1:20],0.05)

#90 km/h speed treatment
AD.90<-quantile(devault_data$AD[21:40],0.05)

#120 km/h speed treatment
AD.120<-quantile(devault_data$AD[41:60],0.05)

#150 km/h speed treatment
AD.150<-quantile(devault_data$AD[61:80],0.05)

#180 km/h speed treatment
AD.180<-quantile(devault_data$AD[81:100],0.05)

#210 km/h speed treatment
AD.210<-quantile(devault_data$AD[101:120],0.05)

#240 km/h speed treatment
AD.240<-quantile(devault_data$AD[121:130],0.05)

#360 km/h speed treatment
AD.360<-quantile(devault_data$AD[131:140],0.05)

#Generating Predicted FIDs for each speed treatment

#60 km/h speed treatment
FIDpredict.60<-output$coef[1]*AD.60
FIDpredict.60
```

```
#90 km/h speed treatment
FIDpredict.90<-output$coef[2]*AD.90
FIDpredict.90

#120 km/h speed treatment
FIDpredict.120<-output$coef[3]*AD.120
FIDpredict.120

#150 km/h speed treatment
FIDpredict.150<-output$coef[4]*AD.150
FIDpredict.150

#180 km/h speed treatment
FIDpredict.180<-output$coef[5]*AD.180
FIDpredict.180

#210 km/h speed treatment
FIDpredict.210<-output$coef[6]*AD.210
FIDpredict.210

#240 km/h speed treatment
FIDpredict.240<-output$coef[7]*AD.240
FIDpredict.240

#360 km/h speed treatment
FIDpredict.360<-output$coef[8]*AD.360
FIDpredict.360


#Assemblying a data frame of the predicted FID at each speed treatment
a<-rbind(60,90,120,150,180,210,240,360)
b<-
matrix(rbind(FIDpredict.60,FIDpredict.90,FIDpredict.120,FIDpredict.150,FIDpredict.180,FIDpredict.210
,FIDpredict.240,FIDpredict.360))
FearFID<-data.frame(cbind(a,b))
colnames(FearFID)=c("Speed","FID")

#Graphing the predict FID for the FEAR hypothesis for each speed treatment
FearFID.graph<-ggplot(data=NULL, aes(x = FearFID[,1], y=FearFID[,2], label= round(FearFID[,2],
2)))+
  geom_text(hjust =-.1, vjust = .5, nudge_x = 0, size=5)+
  geom_smooth(method="lm")+
  geom_point(size=2)+
  labs(y = "Predicted FID (m)",x = " Vehicle Speed (km/h)")+
  theme_classic(base_size = 16)+
  scale_x_continuous(breaks=c(0, 50, 100, 150, 200, 250, 300, 350))+
  ylim(0,30)

FearFID.graph
```

```
ADFIDslope<-lm(FID~Speed, data = FearFID)
summary(ADFIDslope)

########################################Looming Stimulus Hypothesis

#Half width of an approaching vehicle
r<-1.725/2

#Creating a new data frame for the Looming Stimulus Hypothesis
devault_data_l<-devault_data

#Converting km/h into meters per second.
devault_data_l$Speed<-devault_data_l$Speed*0.27778

#Estimating the physiological delay between the onset of neurons firing and when they neurons reach peak firing rate
#triggering an escape response following procedures in Fotowat, H., & Gabbiani, F. (2011).
#The average TTC Flight per each speed treatment in DeVault et al.2015
TTCF.60<-mean(devault_data_l$TTCF[1:20])
TTCF.90<-mean(devault_data_l$TTCF[21:30])
TTCF.120<-mean(devault_data_l$TTCF[41:50])
TTCF.150<-mean(devault_data_l$TTCF[61:70])
TTCF.180<-mean(devault_data_l$TTCF[81:90])
TTCF.210<-mean(devault_data_l$TTCF[101:110])
TTCF.240<-mean(devault_data_l$TTCF[121:130])
TTCF.360<-mean(devault_data_l$TTCF[131:140])

### calculating the Ratio of Size to Speed
ratio.60<-r/devault_data_l$Speed[1]
ratio.90<-r/devault_data_l$Speed[21]
ratio.120<-r/devault_data_l$Speed[41]
ratio.150<-r/devault_data_l$Speed[61]
ratio.180<-r/devault_data_l$Speed[81]
ratio.210<-r/devault_data_l$Speed[101]
ratio.240<-r/devault_data_l$Speed[121]
ratio.360<-r/devault_data_l$Speed[131]

#Creating a data frame with the mean TTCFlight (i.e., the amount of seconds remaining prior to collision at which the animal escaped)
#and the ratio of approach speed and size, a proxy for visual angle and the rate of change in the visual angle.
a<-matrix(rbind(TTCF.60,TTCF.90,TTCF.120,TTCF.150,TTCF.180,TTCF.210,TTCF.240,TTCF.360))
b<-matrix(rbind(ratio.60,ratio.90,ratio.120,ratio.150,ratio.180,ratio.210,ratio.240,ratio.360))

df<-data.frame(cbind(a,b))
colnames(df)<-c("TTCFlight","Ratio")
df

#Graphing the relationship between TTCFlight and the ratio between size and approach speed
looming.graph<-ggplot(data=df, aes(x = Ratio, y= TTCFlight))+
  #geom_text(hjust = 0.05, vjust = -0.5, nudge_x = -0.05, size=4)+
```

```
  geom_smooth(method="lm", formula= y~x)+ geom_point(size=3)+
  labs(y = "Time to Collision Flight (s)",x = " Ratio of Size to Speed ")+
  theme_classic(base_size = 16)+
  ylim(0,2)+
  xlim(0,0.15)
looming.graph

###The intercept estimates the parameter in the looming model for the physiological delay between the onset of neuron firing
#and when neurons reach their peak firing rate
loom_model<-lm(TTCFlight~Ratio, data = df)
summary(loom_model)


##########################Critical Angle for 60
#vehicle size(object size)
delay<-loom_model$coefficients[1]

#Putting vehicle speed in m/s into an object

#ie vehicle speed 60km/h
v.60<- devault_data_l$Speed[1]

#ie vehicle speed 90km/h
v.90<- devault_data_l$Speed[21]

#ie vehicle speed 120km/h
v.120<- devault_data_l$Speed[41]

#ie vehicle speed 150km/h
v.150<- devault_data_l$Speed[61]

#ie vehicle speed 180km/h
v.180<- devault_data_l$Speed[81]

#ie vehicle speed 210km/h
v.210<- devault_data_l$Speed[101]

#ie vehicle speed 240km/h
v.240<- devault_data_l$Speed[121]

#ie vehicle speed 360km/h
v.360<- devault_data_l$Speed[131]

###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###60 km/h speed treatment
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.60<-2*(atan((r/(v.60*TTCF.60)))*(180/pi))
angle.FID.60

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
```

```
angle.expand.60<- (1/(((v.60/(r*2))*(TTCF.60^2))+((r*2)/(4*v.60))))
angle.expand.60

##### Calculates Tau Ratio
tau.deg.60<-angle.FID.60/angle.expand.60
tau.deg.60

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.60<-(r/(tan((angle.FID.60/2)*(pi/180)))) - (v.60*delay)
FID.60


###90 km/h speed treatment
###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.90<-2*(atan((r/(v.90*TTCF.90)))*(180/pi))
angle.FID.90

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
angle.expand.90<- (1/(((v.90/(r*2))*(TTCF.90^2))+((r*2)/(4*v.90))))
angle.expand.90

##### Calculates Tau Ratio
tau.deg.90<-angle.FID.90/angle.expand.90
tau.deg.90

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.90<-(r/(tan((angle.FID.90/2)*(pi/180)))) - (v.90*delay)
FID.90



###120 km/h speed treatment
###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.120<-2*(atan((r/(v.120*TTCF.120)))*(180/pi))
angle.FID.120

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
angle.expand.120<- (1/(((v.120/(r*2))*(TTCF.120^2))+((r*2)/(4*v.120))))
angle.expand.120

##### Calculates Tau Ratio
tau.deg.120<-angle.FID.120/angle.expand.120
tau.deg.120

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.120<-(r/(tan((angle.FID.120/2)*(pi/180)))) - (v.120*delay)
```

```
FID.120


###150 km/h speed treatment
###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.150<-2*(atan((r/(v.150*TTCF.150)))*(180/pi))
angle.FID.150

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
angle.expand.150<- (1/(((v.150/(r*2))*(TTCF.150^2))+((r*2)/(4*v.150))))
angle.expand.150

##### Calculates Tau Ratio
tau.deg.150<-angle.FID.150/angle.expand.150
tau.deg.150

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.150<-(r/(tan((angle.FID.150/2)*(pi/180)))) - (v.150*delay)
FID.150


###180 km/h speed treatment
###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.180<-2*(atan((r/(v.180*TTCF.180)))*(180/pi))
angle.FID.180

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
angle.expand.180<- (1/(((v.180/(r*2))*(TTCF.180^2))+((r*2)/(4*v.180))))
angle.expand.180

##### Calculates Tau Ratio
tau.deg.180<-angle.FID.180/angle.expand.180
tau.deg.180

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.180<-(r/(tan((angle.FID.180/2)*(pi/180)))) - (v.180*delay)
FID.180


###210 km/h speed treatment
###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.210<-2*(atan((r/(v.210*TTCF.210)))*(180/pi))
angle.FID.210

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
angle.expand.210<- (1/(((v.210/(r*2))*(TTCF.210^2))+((r*2)/(4*v.210))))
```

```
angle.expand.210

##### Calculates Tau Ratio
tau.deg.210<-angle.FID.210/angle.expand.210
tau.deg.210

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.210<-(r/(tan((angle.FID.210/2)*(pi/180)))) - (v.210*delay)
FID.210


###240 km/h speed treatment
###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.240<-2*(atan((r/(v.240*TTCF.240)))*(180/pi))
angle.FID.240

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
angle.expand.240<- (1/(((v.240/(r*2))*(TTCF.240^2))+((r*2)/(4*v.240))))
angle.expand.240

##### Calculates Tau Ratio
tau.deg.240<-angle.FID.240/angle.expand.240
tau.deg.240

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.240<-(r/(tan((angle.FID.240/2)*(pi/180)))) - (v.240*delay)
FID.240

###360 km/h speed treatment
###Trigonometry in R is in radians, (180/pi) converts radians -> degrees
###Calculates the visual angle of the approaching vehicle at TTC Flight
angle.FID.360<-2*(atan((r/(v.360*TTCF.360)))*(180/pi))
angle.FID.360

###Calculates the rate at which the visual angle expands for the approaching vehicle at TTC Flight
angle.expand.360<- (1/(((v.360/(r*2))*(TTCF.360^2))+((r*2)/(4*v.360))))
angle.expand.360

##### Calculates Tau Ratio
tau.deg.360<-angle.FID.360/angle.expand.360
tau.deg.360

#Predicted FID according to the hypothesis
#The visual angle essentially predicts the FID based on TTC and then accounts for the neuronal latency
FID.360<-(r/(tan((angle.FID.360/2)*(pi/180)))) - (v.360*delay)
FID.360

####Putting The outputs into a single data frame
```

```
a<-matrix(rbind(TTCF.60,TTCF.90,TTCF.120,TTCF.150,TTCF.180,TTCF.210,TTCF.240,TTCF.390))
b<-matrix(rbind(FID.60,FID.90,FID.120,FID.150,FID.180,FID.210,FID.240,FID.360))
c<-
matrix(rbind(tau.deg.60,tau.deg.90,tau.deg.120,tau.deg.150,tau.deg.180,tau.deg.210,tau.deg.240,tau.deg.3
60))
d<-
matrix(rbind(angle.FID.60,angle.FID.90,angle.FID.120,angle.FID.150,angle.FID.180,angle.FID.210,angl
e.FID.240,angle.FID.360))
e<-
matrix(rbind(angle.expand.60,angle.expand.90,angle.expand.120,angle.expand.150,angle.expand.180,ang
le.expand.210,angle.expand.240,angle.expand.360))
f<-matrix(rbind(60,90,120,150,180,210,240,360))

df.1<-data.frame(cbind(a,b,c,d,e,f))
colnames(df.1)<-c("TTCF","FID","Tau","FID.Angle","Expansion.Angle","Speed")

mean(df.1$FID)

####How the predicted FID Changes with approach Speed
FID.Speed<-ggplot(data=df.1, aes(x = Speed, y=FID, label=round(FID,2)))+
  geom_smooth(method="lm", formula= y~x)+ geom_point(size=3)+
  labs(y = "Predicted FID (m)",x = "Vehicle Speed (km/h)")+
  geom_text(hjust = 0.4, vjust = -1.25, nudge_x = -0.05, size=4)+
  scale_x_continuous(breaks=c(0, 50, 100, 150, 200, 250, 300, 350))+theme_classic(base_size = 16)
+ylim(0,45)

FID.Speed
mean(df.1$FID)

#Model Predicted FID with Approach Speed, Approach Speed was run as a quantitative variable in the
model
loom_model<-lm(FID~Speed, data = df.1)
summary(loom_model)



############################Visual Cue Model

#The mean alert distance per each speed treatment
AD.60<-mean(devault_data$AD[1:20])
AD.90<-mean(devault_data$AD[21:40])
AD.120<-mean(devault_data$AD[41:60])
AD.150<-mean(devault_data$AD[61:80])
AD.180<-mean(devault_data$AD[81:100])
AD.210<-mean(devault_data$AD[101:120])
AD.240<-mean(devault_data$AD[121:130])
AD.360<-mean(devault_data$AD[131:140])

#The mean flight initiation distance per each speed treatment
FID.60<-round(mean(devault_data$FID[1:20]),0)
FID.90<-round(mean(devault_data$FID[21:40]),0)
```

```
FID.120<-round(mean(devault_data$FID[41:60]),0)
FID.150<-round(mean(devault_data$FID[61:80]),0)
FID.180<-round(mean(devault_data$FID[81:100]),0)
FID.210<-round(mean(devault_data$FID[101:120]),0)
FID.240<-round(mean(devault_data$FID[121:130]),0)
FID.360<-round(mean(devault_data$FID[131:140]),0)

#e<-data.frame(AD.60,AD.90,AD.120,AD.150,AD.180,AD.210,AD.240,AD.360)
#e<-data.frame(stack(e))

#Parameters
#Profile Size, Diameter of the Vehicle
A<-(1.725)*pi
#Approach Angle
a<-0.00
#Vegtation Parameter
c<-0
#Maximum Possible Detection distance
detection<-dd
#Change in that distance as the vehicle moves closer
delta<-seq(474,1,by=-1)

####Estimated change in perceived vehicle size when viewed at two distances

#The first perceived visual cue based on profile size
view_1<-A/(detection^2)

#The second perceived visual cue based on profile size
view_2<-A/((detection-delta)^2)

#The change in profile size at every distance from the approaching vehicle
delta_A<-(view_1-view_2)
#plot(deltaA)

#Estimation of the threshold change in perceived profile size which triggers an escape response
#based on the mean FID for each speed treatment
threshold_delta_A.60<-delta_A[FID.60]

threshold_delta_A.90<-delta_A[FID.90]
threshold_delta_A.120<-delta_A[FID.120]
threshold_delta_A.150<-delta_A[FID.150]
threshold_delta_A.180<-delta_A[FID.180]
threshold_delta_A.210<-delta_A[FID.210]
threshold_delta_A.240<-delta_A[FID.240]
threshold_delta_A.360<-delta_A[FID.360]


#The function to solve for the predicted FID
#AD, the second distance at which prey receive a visual cue,
bisection<-function(f, a, b, num = 10, eps = 1e-05)
{
```

```
  h = abs(b - a)/num
  i = 0
  j = 0
  a1 = b1 = 0
  while (i <= num) {
    a1 = a + i * h
    b1 = a1 + h
    if (f(a1) == 0) {
      #print(a1)
      #print(f(a1))
    }
    else if (f(b1) == 0) {
      print(b1)
      print(f(b1))
    }
    else if (f(a1) * f(b1) < 0) {
      repeat {
        if (abs(b1 - a1) < eps)
          break
        x <- (a1 + b1)/2
        if (f(a1) * f(x) < 0)
          b1 <- x
        else a1 <- x
      }
      #print(j + 1)
      j = j + 1
      #print((a1 + b1)/2)
      #print(f((a1 + b1)/2))
      return(a1)
    }
    i = i + 1
  }
}

###########Predicted FID Visual Cue Model

#60 km/h speed treatment
vc.60<-function(FID_vc) {
  with (as.list(params),{
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.60, delta_A=threshold_delta_A.60, A=A)

FID_60_vc<-bisection(vc.60,0,AD.60)


#90 km/h speed treatment
vc.90<-function(FID_vc) {
  with (as.list(params),{
```

```
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.90, delta_A=threshold_delta_A.90, A=A)

FID_90_vc<-bisection(vc.90,0,AD.90)

#120 km/h speed treatment
vc.120<-function(FID_vc) {
  with (as.list(params),{
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.120, delta_A=threshold_delta_A.120, A=A)

FID_120_vc<-bisection(vc.120,0,AD.120)

#150 km/h speed treatment
vc.150<-function(FID_vc) {
  with (as.list(params),{
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.150, delta_A=threshold_delta_A.150, A=A)

FID_150_vc<-bisection(vc.150,0,AD.150)

#180 km/h speed treatment
vc.180<-function(FID_vc) {
  with (as.list(params),{
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.180, delta_A=threshold_delta_A.180, A=A)

FID_180_vc<-bisection(vc.180,0,AD.180)

#210 km/h speed treatment
vc.210<-function(FID_vc) {
  with (as.list(params),{
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.210, delta_A=threshold_delta_A.210, A=A)

FID_210_vc<-bisection(vc.210,0,AD.210)
```

```
#240 km/h speed treatment
vc.240<-function(FID_vc) {
  with (as.list(params),{
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.240, delta_A=threshold_delta_A.240, A=A)

FID_240_vc<-bisection(vc.240,0,AD.240)

#360 km/h speed treatment
vc.360<-function(FID_vc) {
  with (as.list(params),{
    df.FID<-(1-c)*((1/((FID_vc-AD*cos(a))^2)+(AD^2*sin(a)^2))-(1/(FID_vc^2)))-delta_A/A

    return(c(df.FID=df.FID))
  })}
params<-c(c=0, a=0, AD=AD.360, delta_A=threshold_delta_A.360, A=A)

FID_360_vc<-bisection(vc.360,0,AD.360)

#Creating a data frame of the predicted FID for the visual cue model
a<-
matrix(rbind(FID_60_vc,FID_90_vc,FID_120_vc,FID_150_vc,FID_180_vc,FID_210_vc,FID_240_vc,FI
D_360_vc))
b<-matrix(rbind(60,90,120,150,180,210,240,360))

df_vc<-data.frame(cbind(a,b))
colnames(df_vc)<-c("FID","Speed")

mean(df_vc$FID)

#Ploting the FID predicted and approach speed for the visualcue model
FID.plot<-ggplot(data=df_vc, aes(x = Speed, y= FID, label=round(FID,2)))+
  geom_smooth(method="lm", formula= y~x)+ geom_point(size=3)+
  labs(y = "Predicted FID (m)",x = "Vehicle Speed (km/h)")+
  geom_text(hjust = -0.4, vjust = 0.15, nudge_x = -0.05, size=4)+
  scale_x_continuous(breaks=c(0, 50, 100, 150, 200, 250, 300, 350))+theme_classic(base_size = 16)
+ylim(0,45)
FID.plot


############################Bayesian optimal escape model


rm(list = ls ())
#
devault_data<-read.csv("DeVaultData.csv", header=TRUE)
```

```
#Creating a new data frame for the Bayesian optimal escape model
devault_data_b<-devault_data

#Converting km/h into meters per second.
devault_data_b$Speed<-devault_data_b$Speed*0.27778

#Mean alert distance for each speed speed treatment form DeVault et al.2015
AD.60<-mean(devault_data$AD[1:20])
AD.90<-mean(devault_data$AD[21:40])
AD.120<-mean(devault_data$AD[41:60])
AD.150<-mean(devault_data$AD[61:80])
AD.180<-mean(devault_data$AD[81:100])
AD.210<-mean(devault_data$AD[101:120])
AD.240<-mean(devault_data$AD[121:130])
AD.360<-mean(devault_data$AD[131:140])

#Function for predicting FID for the Bayesian optimal escape model from Sutton & O'Dwyer 2018


bayesian<-function(AD,dr,g){
  M<-exp(g)
  d<-seq(0,AD, by=1)
  E<- -.797+(0.659*(log(M)))
  watts<-61.718*((M/1000)^0.7902)
  B<-watts*1.5/1000
  m<-15.9*((M/1000)^0.13)
  h=0.5

  rfs<-function(AD,dr,m){
    rf1<-ifelse(1-d/AD<=1,(1-(d/AD)),0)
    rf2<-ifelse(dr/m<=1,(dr/m),1)
    rf1*rf2
  }

  Energy<-function(E,h,AD,dr,m){((E*((rfs(AD,dr,m)*h)/((rfs(AD,dr,m)*h)+
                                                  ((1-(rfs(AD,dr,m)))*(1-h))))))}

  risk<-data.frame(Energy(E,h,AD,dr,m))
  risk$dist<-d
  colnames(risk)<-c("DEE","distance")
  risk

  a<-risk$distance[which.max(risk$DEE<B)]

  output<-ifelse(a==0,
               ifelse(max(risk$DEE)<B,1,
                      ifelse( min(risk$DEE)>=B,AD,
                      )),a)

  output
}
```

```
#60 km/h speed treatment
output.60<-bayesian(AD.60,devault_data_b$Speed[1],log(43.9))

#90 km/h speed treatment
output.90<-bayesian(AD.90,devault_data_b$Speed[21],log(43.9))

#120 km/h speed treatment
output.120<-bayesian(AD.120,devault_data_b$Speed[41],log(43.9))

#150 km/h speed treatment
output.150<-bayesian(AD.150,devault_data_b$Speed[61],log(43.9))

#180 km/h peed treatment
output.180<-bayesian(AD.180,devault_data_b$Speed[81],log(43.9))

#210 km/h peed treatment
output.210<-bayesian(AD.210,devault_data_b$Speed[101],log(43.9))

#240 km/h peed treatment
output.240<-bayesian(AD.240,devault_data_b$Speed[121],log(43.9))

#360 km/h peed treatment
output.360<-bayesian(AD.240,devault_data_b$Speed[131],log(43.9))


#Creating a data frame of the predicted FID for the Bayesian optimal escape model
output<-
data.frame(rbind(output.60,output.90,output.120,output.150,output.180,output.210,output.240,output.360
))
output$Speed<-c(60,90,120,150,180,210,240,360)
colnames(output)<-c("FID","Speed")
output

#Plotting the relationship between predicted FID and approach speed
graph<-ggplot(data=output, aes(x = Speed, y=FID, label= round(FID,2)))+
  geom_text(hjust = -0.25, vjust = 0.1, nudge_x = -0.05, size=5)+
  geom_smooth(method="lm", formula= y~x)+ geom_point(size=2)+
  labs(y = "Predicted FID (m)",x = "Vehicel Speed (km/h)")+
  theme_classic(base_size = 16)+
  scale_x_continuous(breaks=c(0, 50, 100, 150, 200, 250, 300, 350))+
  ylim(0,55)
graph
```

S.5 part 2. Code for the sensitivity to approach speed evaluation.

```
############################Simulation
rm(list = ls ())


#Packages
library(dplyr)
library(broom)
library(lme4)
library(ggplot2)
library(plotly)
library(plyr)
library(scatterplot3d)

getwd()
setwd("/Users/Ryan/Desktop/Review_MS")
###############Review
#Evaluating the FID and speed relationship for different species based on empirical data from the literature

data<- read.csv("FID_Speed_Review.csv",na.strings = c("","NA"), header=T)

df<-data.frame(data$Species,data$n,data$Speeds,data$FID,data$Stimulus.Type)
colnames(df)<-c("Species","n","Speed","FID","stimulus")

#Estimating the slope and intercept for the FID and approach speed relationship for each species
models <- dlply(df, "Species", function(df)
  lm(FID ~ Speed, data = df))

#Creating a data frame with the slope and intercept for the FID and approach speed relationship for each species
df_slope<-data.frame(ldply(models, coef))
df_slope

#calculating the average FID for each species
a<-data.frame(aggregate(df$FID,list(Species=df$Species), mean))

#adding the mean FID to data frame of slope and intercept
df_slope$FID<-a$x

#adding mass values for each species to the data frame
mass_df<- read.csv("species_mass.csv",na.strings = c("","NA"), header=T)
df_slope$mass<-mass_df$mass

#Estimating Alert distances based on body mass
AD_func<-function(b){
  (10^((0.347*log10(b))))+mean(df_slope$X.Intercept.)
}

min(AD_sim_out[[17]])
```

```
#Adding alert distances to the data frame
df_slope$AD<-as.numeric(AD_func(df_slope$mass))

#Inserting empirically observed alert distances
df_slope$AD<-ifelse(df_slope$Species=="Brown-headed Cowbird",44.65,
                    ifelse(df_slope$Species=="House Sparrow",12.5,
                           ifelse(df_slope$Species=="European Goldfinch",13.7,
                                  ifelse(df_slope$Species=="Hadeda ibis",9.9,df_slope$AD))))


#Plotting the histogram for slope of FID and approach speed for each species
ggplot(data=df_slope, aes(x=Speed))+
  geom_histogram(binwidth=10,fill="white",color="black")+
  xlab("Slope")+ylab("Count")+
  scale_y_continuous(expand =c(0,0))+coord_cartesian(ylim = c(0,30))+
  ggtitle("   Observed Slope of FID & approach speed for 50 Species")+
  theme_classic(base_size = 14)

#Plotting the histogram for Intercept of FID and approach speed for each species
ggplot(data=df_slope, aes(x=X.Intercept.))+
  geom_histogram(binwidth=25,fill="white",color="black")+
  xlab("Intercept")+ylab("Count")+
  scale_y_continuous(expand =c(0,0))+coord_cartesian(ylim = c(0,20))+
  ggtitle("Observed Intercept of FID & approach speed for 50 Species")+
  theme_classic(base_size = 14)

#Plotting the histogram for mean FID for each species
ggplot(data=df_slope, aes(x=FID))+
  geom_histogram(binwidth=10,fill="white",color="black")+
  xlab("Flight Initiation Distance (m)")+ylab("Count")+
  scale_y_continuous(expand =c(0,0))+coord_cartesian(ylim = c(0,15))+
  ggtitle("        Observed Flight Initiation Distance for 50 Species")+
  theme_classic(base_size = 14)

#Plotting the histogram for mean AD for each species
ggplot(data=df_slope, aes(x=AD))+
  geom_histogram(binwidth=5,fill="white",color="black")+
  xlab("Alert Distance (m)")+ylab("Count")+
  scale_y_continuous(expand =c(0,0))+coord_cartesian(ylim = c(0,25))+
  ggtitle("         Estimated & Observed Alert Distance for 50 Species")+
  theme_classic(base_size = 14)


#Renaming columns
colnames(df_slope)<-c("Species","Intercept","Slope","FID","mass","AD")

#Removing species without mass values
df_slope<-na.omit(df_slope)
```

```
###############Simulation

#Parameters

#range of slopes used
slope<-seq(-37,32,by=1)

#range of approach speeds in m/s
x<-seq(1,100,by=1)

#The range of different neuronal latency values used
delay<-seq(.050,.100,length.out = 25)

#Establishing range of body mass values in even intervals along a log scale
mass_df<-df_slope[order(df_slope$mass),]
mass_df_log<-log(mass_df$mass)
mass_df_log<-data.frame(mass_df_log)

df<-matrix(data=0, nrow=50,ncol=1)
df

for(i in 2:50){
  df[i]<- mass_df_log[i,] - mass_df_log[i-1,]
}
df
#Interval of log transformed body mass
diff<-mean(df[2:50,])

mass<-seq(min(mass_df_log),max(mass_df_log),by=diff)


#Functions

#AD Function
AD_function<-function(a,sa) {
  #a is the FID fed to generate the AD, a is essentially the minimum value AD can be
  b<-round(rnorm(1,mean=a,sd=sa))
  AD<-ifelse(b<=0,0,b)
  AD
}

AD_function(mean(df_slope$AD),sd(df_slope$AD))

#ad<-AD_function(mean(df_slope$AD),sd(df_slope$AD))
#ad
###FID function
FID_function<-function(AD,x,m,b,s) {
  #x is speed m/s
  #m is slope
  #b is the intercept
```

```
  #s is the variation in FID
  #The equation for a line + or - some random integer
  #The random integer is being pulled from a uniform distribution
  a<-((rnorm(1,mean=m*x,sd=s)))+b
  ####Limiting extremely large FID values 542, is the maximum distance a cowbird could detect
  #### an approaching vehicle.0 because you can not have a negative FID
  FID<-ifelse(a>=AD,AD,ifelse(a<=0,0,a))
  FID<-round(FID)
  FID
}


#Function used to simulate pairs of AD and FID

sim_function_FID<-function(b,s){
  #x looping through all approach speeds values
  x<-seq(1,100,by=1)
  #m looping through all the different range of slopes
  m<-seq(-37,32,by=1)
  #Making an empty matrix to store AD values
  AD_results <- matrix(nrow= length(x) , ncol= length(m))


  #Simulating our AD's feeding. I am feeding our AD function our simulated FID values
  for(i in 1:length(x)){
    for (j in 1:length(m)){
      AD_results[i,j] <- AD_function(mean(df_slope$AD),sd(df_slope$AD))
    }
  }

  #Making an empty matrix to store FID values
  FID_results <- matrix(nrow= length(x) , ncol= length(m))
  #Simulating our AD's feeding. I am feeding our AD function our simulated FID values
  for(i in 1:length(x)){
    for (j in 1:length(m)){
      FID_results[i,j] <-
FID_function(AD_results[i,j],x[i],m[j],mean(df_slope$Intercept),sd(df_slope$FID))
      #FID_function(results[i,j],x[i],t)
    }
  }




  output = list('AD'=AD_results,'FID'=FID_results)
}

#Repeat the Simulate 7,000 pairs of AD and FID values for 100 iterations
FID_sim_1<-replicate(100,sim_function_FID(mean(df_slope$Intercept),sd(df_slope$FID)),simplify = F)

#Parsing out the FID values
```

```
FID_sim_out<-rep(list(matrix(NA, nrow=100,ncol=70)), 100)

for(i in 1:100) {
  FID_sim_out[[i]]<-FID_sim_1[[i]]$FID   # Printing some output
}

#Parsing out the AD values
AD_sim_out<-rep(list(matrix(NA, nrow=100,ncol=70)), 100)
for(i in 1:100) {
  AD_sim_out[[i]]<-FID_sim_1[[i]]$AD   # Printing some output
}



#Function used to estimate f^2 values
f_func<-function(x){
  f<-(x/(1-x))
}

###FEAR Hypothesis
#Evaluating simulated AD and FID with the FEAR Hypothesis

#Function used to estimate the phi-index value and significance according to the FEAR hypothesis from
Samia & Blumstein 2014

FEAR<-function(AD,FID,N,S){

  phi = numeric(0)

  for(i in 1:S){
    sFID = runif(N,0,AD)                    #simulates random FIDs
    phi[i] = 1 - (sum((AD-sFID)/AD)/N)}

  obs.phi= 1 - (sum((AD-FID)/AD)/ N)   #extract observed phi
  P  = sum(phi >= obs.phi)/(S)              #calculates the P-value of observed phi

  FID_p<-as.numeric(obs.phi*quantile(AD, c(0.05))) #generates predicted FID

  output = list('phi index '= obs.phi,'P-value'=P,'Predicted_FID_95' = FID_p)
  #FID_p
  output
}

#Creating the matrix to store the results of the FEAR hypothesis for each iteration
results_f <- rep(list(matrix(NA, nrow=length(slope),ncol=4)), 100)



AD_df<-data.frame(do.call(rbind,AD_sim_out))
write.csv(AD_df,"Alert_Distance_Simulated_Data.csv")
```

```
FID_df<-data.frame(do.call(rbind,FID_sim_out))
write.csv(FID_df,"Flight_Initiation_Distance_Simulated_Data.csv")

#Evaluating each iteration of the simulation according ot the FEAR hypothesis and parsing the results
into a matrix
for (k in 1:100){
  for (i in 1:length(slope)){
    #Predicted FID
    results_f[[k]][i,1] <-
round(as.numeric(FEAR(AD_sim_out[[k]][,i],FID_sim_out[[k]][,i],length(x),100)[3]))
    #P-value according to the Phi index
    results_f[[k]][i,2] <- as.numeric(FEAR(AD_sim_out[[k]][,i],FID_sim_out[[k]][,i],length(x),100)[2])
    #Phi index  value
    results_f[[k]][i,3] <- as.numeric(FEAR(AD_sim_out[[k]][,i],FID_sim_out[[k]][,i],length(x),100)[1])
    results_f[[k]][i,4]<-slope[i]
    colnames(results_f[[k]])<-c("Predicted FID","P-value","phi","Slope")

  }
}
results_f[1]

#converting the results store in a matrix into a data frame
fear_df<-data.frame(do.call(rbind,results_f))

head(fear_df)
nrow(fear_df)
#writing out the results for the FEAR hypothesis for every simulation
write.csv(fear_df, file="Fear_Results.csv")

#reading in the results of every simulation back in
#fear_df<-read.csv("fear_model.csv")

#creating the matrix to store the average results of the FEAR hypothesis for each slope
eval_f<-matrix(NA, nrow=70,ncol=3)
eval_f

fear_df

#Function used to estimate the mean phi value and variation in phi value for each slope
eval_function_f<-function(df,y){

  a<-ifelse(df[,4]==y,df[,3],NA)
  #a<-ifelse(df[,4]==y,df[,1],NA)

  eval_df<-data.frame(a)
  eval_df<-na.omit(eval_df)

  colnames(eval_df)<-c("phi")

  output = list("mean_phi"=mean(eval_df$phi), "phi_SD"=sd(eval_df$phi), y)
```

```
  output
}

#eval_function_f(fear_df,slope[1])

#Calculating the mean phi value and standard deviation for each slope and parsing the results into a
matrix
for (i in 1:length(slope)){
  #mean phifor each slope
  eval_f[i,1]<-as.numeric(eval_function_f(fear_df,slope[i])[[1]])
  #variation in phi for each slope
  eval_f[i,2]<-as.numeric(eval_function_f(fear_df,slope[i])[[2]])
  #slope
  eval_f[i,3]<-as.numeric(eval_function_f(fear_df,slope[i])[[3]])
}

#Renaming the column names of the matrix
colnames(eval_f)<-c("mean_phi","sd_phi","slope")

#Reading out the results for each slope
write.csv(eval_f, file="FEAR_Results_Summary.csv")


eval_f<-data.frame(eval_f)

#Plotting the mean phi value for each slope
ggplot(data=eval_f,aes(x=slope, y=mean_phi)) +
  geom_point(size=2)+
  geom_line(lwd=2)+
  xlab("Slope") +
  ggtitle("                              FEAR Hypothesis")+
  ylab("Phi Index")+
  scale_y_continuous(expand =c(0,0))+coord_cartesian(ylim = c(0,1.1))+
  theme_classic(base_size = 16)

#Mean Mean Phi and SD value for negative slopes
mean(eval_f[1:37,1])
sd(eval_f[1:37,1])

#Mean Mean Phi and SD value for positive slopes
mean(eval_f[39:70,1])
sd(eval_f[39:70,1])

###Looming stimulus hypothesis

#Function used to generate predicted FID according to the looming stimulus hypothesis

Looming<-function(FID,AD,delay,Speed){
  #width of vehicle
  r=1.725
  #Estimating the TTC Flight
```

```
  TTCF<-(FID/Speed)
  #Generating predicted FID according to the Looming stimulus hypothesis
  FID_p<-ifelse((FID-(Speed*delay))<=0,0,
              ifelse(FID-(Speed*delay)>=AD,AD,
                   round(FID-(Speed*delay),2)))

  #Calculating the visual angle of the approaching vehicle when the animal escapes
  angle.FID<-2*(atan((r/(FID_p)))*(180/pi))

  #Calculates the rate at which the visual angle expands for the approaching vehicle when the animal
escapes
  angle.expand<- (1/(((Speed/(r*2))*(TTCF^2))+((r*2)/(4*Speed))))
  #angle.expand

  #Estimates the Tau Ratio when the animal escapes
  tau.deg<-angle.FID/angle.expand

  #Output is the predicted FID
  FID_p

}



#Creating the matrix to store the predicted FID values for the looming stimulus hypothesis
results_l<-rep(list(rep(list(matrix(NA, nrow=100,ncol=70)), 25)),100)

#Using the simulated FID and AD pairs to generate predicted FID for the looming stimulus hypothesis
and parsing the results
#for each simulation into a matrix
for(k in 1:100){
  for(i in 1:length(x)){
    for (j in 1:length(slope)){
      for (l in 1:length(delay)){
        results_l[[k]][[l]][i,j] <- Looming(FID_sim_out[[k]][i,j],AD_sim_out[[k]][i,j],delay[l],x[i])
      }
    }
  }
}




#Turning the matrix into a data frame
list <- unlist(results_l, recursive = FALSE)
loom_df <- do.call("rbind",list)
nrow(loom_df)
2+2
results_l_df<-data.frame(do.call(rbind,results_l))
results_l[[1]]
```

```
head(results_l_df)

#Reading out the model predicted FID for the looming stimulus hypothesis
write.csv(loom_df, file="Looming_FID_Data.csv")


#Creating a matrix to store the the evaluation results for the looming stimulus hypothesis
effect_l_size<-rep(list(rep(list(matrix(NA, nrow=70,ncol=5)),25)),100)

##Evaluating the model predicted FID for the looming stimulus hypothesis and parsing the results in a
matrix

for(k in 1:100){
  for (l in 1:length(delay)){
    for (j in 1:length(slope)){
      effect_l_size[[k]][[l]][j,1]<-summary(lm(results_l[[k]][[l]][,j]~x))$adj.r.squared
      effect_l_size[[k]][[l]][j,2]<-summary(lm(results_l[[k]][[l]][,j]~x))$r.squared
      effect_l_size[[k]][[l]][,3]<-delay[l]
      effect_l_size[[k]][[l]][,4]<-slope
      effect_l_size[[k]][[l]][,5]<-k
      colnames(effect_l_size[[k]][[l]])<-c("Adj_r_squared","r_squared","delay","slope","simulation")
    }
  }
}

#Converting matrix into data frame
list <- unlist(effect_l_size, recursive = FALSE)
loom_df <- do.call("rbind",list)
head(loom_df)
tail(loom_df)
write.csv(loom_df, file="Looming_Results.csv")

#Function used to estimate the mean R_squared between FID and approach for each slope in the Looming
Stimulus hypothesis
eval_function_l<-function(df,y,x){

  a<-ifelse(df[,4]==y,
          ifelse(df[,3]==x,df[,1],NA),NA)
  b<-ifelse(df[,4]==y,
          ifelse(df[,3]==x,df[,2],NA),NA)

  eval_df<-data.frame(cbind(a,b))
  eval_df<-na.omit(eval_df)

  colnames(eval_df)<-c("Adj_R_Squared","R_squared")

  output = list('Adj_R_squared'= mean(eval_df$Adj_R_Squared),'Adj_R_Squared_SD'=
sd(eval_df$Adj_R_Squared),
              'R_Squared'=mean(eval_df$R_squared),'R_squared_SD'= sd(eval_df$R_squared),
              y,x)
```

```
  #output = list(mean(eval_df[,2]),sd(eval_df[,2]),mean(eval_df[,3]),
  #    sd(eval_df[,3]),f,x)

  output
}


#Creating the matrix to store the R_squared results for each slope for the Looming Stimulus Hypothesis
eval_l<-rep(list(matrix(NA, nrow=70,ncol=6)),25)


#Calculating the mean R_squared value and standard deviation in R_squared for each slope and parsing the results into a matrix
for(j in 1:length(delay)){
  for (i in 1:length(slope)){
    #Adj.R_squared
    eval_l[[j]][i,1]<-as.numeric(eval_function_l(loom_df,slope[i],delay[j])[[1]])
    eval_l[[j]][i,2]<-as.numeric(eval_function_l(loom_df,slope[i],delay[j])[[2]])
    #R_Squared
    eval_l[[j]][i,3]<-as.numeric(eval_function_l(loom_df,slope[i],delay[j])[[3]])
    eval_l[[j]][i,4]<-as.numeric(eval_function_l(loom_df,slope[i],delay[j])[[4]])
    #Slope
    eval_l[[j]][i,5]<-as.numeric(eval_function_l(loom_df,slope[i],delay[j])[[5]])
    #Delay
    eval_l[[j]][i,6]<-as.numeric(eval_function_l(loom_df,slope[i],delay[j])[[6]])

  }
}

#Converting the matrix into a data.frame
eval_l_df<-data.frame(do.call(rbind,eval_l))

#Renaming the columns of the matrix
colnames(eval_l_df)<-c('Adj_R_squared','Adj_R_squared_SD',
                       'R_squared','R_squared_SD',
                       'Slope','Delay')
write.csv(eval_l_df, file="Looming_Results_Summary.csv")
#######Plotting the results of the looming stimulus hypothesis

eval_l_df_2D<-eval_l_df
#2D plotting of looming stimulus hypothesis
#Only including a single neuronal latency value, 0.075 sec

eval_l_df_2D$Delay<-round(eval_l_df$Delay,3)

eval_l_df_2D$Slope<-(ifelse(eval_l_df_2D$Delay==0.075,eval_l_df_2D$Slope,NA))
eval_l_df_2D<-na.omit(eval_l_df_2D)

#Estimating f_squared from Cohen 1988 from R_squared
f<-lapply(eval_l_df_2D$R_squared, f_func)
f_squared<-do.call(rbind,f)
```

```
#appending f_squared to data frame
eval_l_df_2D$f_squared<-f_squared

#2D graph
ggplot(data=eval_l_df_2D,aes(x=Slope, y=f_squared)) +
  geom_point(size=2)+
  geom_line(lwd=2)+
  xlab("Slope") +
  ggtitle("                                   Looming Stimulus Hypothesis")+
  ylab(expression(italic(f^{2})))+
  scale_y_continuous(expand =c(0,0))+coord_cartesian(ylim = c(0,.65))+
  theme_classic(base_size = 16)

#Mean Mean f^2 and SD value for negative slopes
mean(eval_l_df_2D[1:37,7])
sd(eval_l_df_2D[1:37,7])

#Mean Mean Phi and SD value for positive slopes
mean(eval_l_df_2D[39:70,7])
sd(eval_l_df_2D[39:70,7])



#Generating a 3D graph between delay (s), slope, and f^2 for the looming stimulus hypothesis
#putting f^2 into a matrix
f_Squared<-matrix(eval_l_df$R_squared,nrow = 70, ncol=25)
f_Squared

#Crating axis titles
axx <- list(
  title = "Neuronal Latency (s)"
)

axy <- list(
  title = "Slope"
)

axz <- list(
  title = "f squared"
)
delay

#Plotting the three different variables
surf_l<-plot_ly(x=~delay,y=~slope, z = ~f_Squared, type = "surface", showlegend=T)
surf_l <- surf_l %>% layout(scene = list(xaxis=axx,yaxis=axy,zaxis=axz))
surf_l <- surf_l %>%layout(radialaxis = list(ticksuffix = "%"), orientation = 270)
surf_l

###Bayesian optimal escape model

#Function used to generate predicted FID according to the Bayesian optimal escape model
```

```
bayesian<-function(AD,g,dr){
 M<-exp(g)
 d<-seq(0,AD, by=1)
 E<- -.797+(0.659*(log(M)))
 watts<-61.718*((M/1000)^0.7902)
 B<-watts*1.5/1000
 m<-15.9*((M/1000)^0.13)
 h=0.5

 rfs<-function(AD,dr,m){
   rf1<-ifelse(1-d/AD<=1,(1-(d/AD)),0)
   rf2<-ifelse(dr/m<=1,(dr/m),1)
   rf1*rf2
 }
 Energy<-function(E,h,AD,dr,m){((E*((rfs(AD,dr,m)*h)/((rfs(AD,dr,m)*h)+
                                              ((1-(rfs(AD,dr,m)))*(1-h))))))}
 risk<-data.frame(Energy(E,h,AD,dr,m))
 risk$dist<-d
 colnames(risk)<-c("DEE","distance")
 risk

 a<-risk$distance[which.max(risk$DEE<B)]

 output<-ifelse(AD>0,
              (ifelse(a==0,
                ifelse(max(risk$DEE)<B,1,
                    ifelse(min(risk$DEE)>=B,AD,
                      )),a)),0)
 output
}
#bayesian(0,5,50)


#x
#mass
#slope
#Creating the matrix to store the predicted FID values for the Bayesian optimal escape model
results_b<-rep(list(rep(list(matrix(NA, nrow=100,ncol=70)), 50)),100)

#results_b[17]

#AD_sim_out[[1]][[]]
#Using the simulated AD to generate predicted FID for the Bayesian optimal escape model and parsing the results
#for each simulation into a matrix
for(k in 1:100){
   for(i in 1:length(x)){
     for (j in 1:length(slope)){
       for (l in 1:length(mass)){
       results_b[[k]][[l]][i,j] <- bayesian(AD_sim_out[[k]][i,j],mass[l],x[i])
     }
```

```
    }
  }
}


list <- unlist(results_b, recursive = FALSE)
bayes_df <- do.call("rbind",list)
head(bayes_df)

write.csv(bayes_df, file="Bayesian_FID_Data.csv")

 #Creating a matrix to store the the evaluation results for the Bayesian optimal escape model
effect_b_size<-rep(list(rep(list(matrix(NA, nrow=70,ncol=5)),50)),100)

#effect_b_size[[17]]
#Evaluating the model predicted FID for the Bayesian optimal escape model and parsing the results in a
matrix

for(k in 1:100){
  for (l in 1:length(mass)){
    for (j in 1:length(slope)){
      effect_b_size[[k]][[l]][j,1]<-summary(lm(results_b[[k]][[l]][,j]~x))$adj.r.squared
      effect_b_size[[k]][[l]][j,2]<-summary(lm(results_b[[k]][[l]][,j]~x))$r.squared
      effect_b_size[[k]][[l]][,3]<-mass[l]
      effect_b_size[[k]][[l]][j,4]<-slope[j]
      effect_b_size[[k]][[l]][,5]<-k
      colnames(effect_b_size[[k]][[l]])<-c("Adj_r_squared","r_squared","mass","slope","simulation")
    }
  }
}


#Converting matrix into data frame
list <- unlist(effect_b_size, recursive = FALSE)
bayesian_df <- do.call("rbind",list)
head(bayesian_df)


write.csv(bayesian_df, file="Bayesian_Results.csv")

#Function used to estimate the mean R_squared between FID and approach for each slope in the Bayesian
optimal escape model
eval_function_b<-function(df,y,x){

  a<-ifelse(df[,4]==y,
           ifelse(df[,3]==x,df[,1],NA),NA)
  b<-ifelse(df[,4]==y,
           ifelse(df[,3]==x,df[,2],NA),NA)

  eval_df<-data.frame(cbind(a,b))
  eval_df<-na.omit(eval_df)
```

```
  colnames(eval_df)<-c("Adj_R_Squared","R_squared")

  output = list('Adj_R_squared'= mean(eval_df$Adj_R_Squared),'Adj_R_Squared_SD'=
sd(eval_df$Adj_R_Squared),
                'R_Squared'=mean(eval_df$R_squared),'R_squared_SD'= sd(eval_df$R_squared),
                x,y)

  #output = list(mean(eval_df[,2]),sd(eval_df[,2]),mean(eval_df[,3]),
  #    sd(eval_df[,3]),f,x)

  output
}


#Creating the matrix to store the R_squared results for each slope for the Bayesian optimal escape model
eval_b<-rep(list(matrix(NA, nrow=70,ncol=6)),50)



#Calculating the mean R_squared value and standard deviation in R_squared for each slope and parsing
the results into a matrix

for(j in 1:length(mass)){
  for (i in 1:length(slope)){
    #Adj.R_squared
    eval_b[[j]][i,1]<-as.numeric(eval_function_b(bayesian_df,slope[i],mass[j])[[1]])
    eval_b[[j]][i,2]<-as.numeric(eval_function_b(bayesian_df,slope[i],mass[j])[[2]])
    #R_Squared
    eval_b[[j]][i,3]<-as.numeric(eval_function_b(bayesian_df,slope[i],mass[j])[[3]])
    eval_b[[j]][i,4]<-as.numeric(eval_function_b(bayesian_df,slope[i],mass[j])[[4]])
    #Mass
    eval_b[[j]][i,5]<-as.numeric(eval_function_b(bayesian_df,slope[i],mass[j])[[5]])
    #Slope
    eval_b[[j]][i,6]<-as.numeric(eval_function_b(bayesian_df,slope[i],mass[j])[[6]])

  }
}

list <- unlist(effect_b_size, recursive = FALSE)
bayesian_df <- do.call("rbind",list)
head(bayesian_df)


write.csv(bayesian_df, file="Bayesian_Results.csv")

#Converting the matrix into a data.frame
eval_b_df<-data.frame(do.call(rbind,eval_b))
head(eval_b_df)
nrow(eval_b_df)
```

```
#Renaming the columns of the matrix
colnames(eval_b_df)<-
c('Adj_R_squared','Adj_R_squared_SD','R_squared','R_squared_SD','Mass','Slope')
eval_b_df$Mass<-round(eval_b_df$Mass,5)

write.csv(eval_b_df, file="Bayesian_Results_Summary.csv")

#######Plotting the results of the Bayesian optimal escape model


#2D plotting of looming stimulus hypothesis
#Only including a single log body mass value value, 0.075 sec
eval_b_df$Mass<-(ifelse(eval_b_df$Mass==5.52511,eval_b_df$Mass,NA))

eval_b_df<-na.omit(eval_b_df)

#Estimating f_squared from Cohen 1988 from R_squared
f<-lapply(eval_b_df$R_squared, f_func)
f_squared<-do.call(rbind,f)
#appending f_squared to data frame
eval_b_df$f_squared<-f_squared

#2D graph
ggplot(data=eval_b_df,aes(x=Slope, y=f_squared)) +
  geom_point(size=2)+
  geom_line(lwd=2)+
  xlab("Slope") +
  ggtitle("                Bayesian optimal escape model")+
  #geom_errorbar(aes(ymin=Adj_R_squared-Adj_R_squared_SD,
ymax=Adj_R_squared+Adj_R_squared_SD), colour="black", width=.1)+
  ylab(expression(italic(f^{2})))+
  scale_y_continuous(expand =c(0,0))+coord_cartesian(ylim = c(0,.15))+
  #ylab("Total Cost of Collision per species (millions)")+
  theme_classic(base_size = 16)


#Mean Mean f^2 and SD value for negative slopes
mean(eval_b_df[1:37,7])
sd(eval_b_df[1:37,7])

#Mean Mean Phi and SD value for positive slopes
mean(eval_b_df[39:70,7])
sd(eval_b_df[39:70,7])



#Converting the matrix into a data.frame
eval_b_df<-data.frame(do.call(rbind,eval_b))
head(eval_b_df)
nrow(eval_b_df)
```

```
#Renaming the columns of the matrix
colnames(eval_b_df)<-
c('Adj_R_squared','Adj_R_squared_SD','R_squared','R_squared_SD','Mass','Slope')

f<-lapply(eval_b_df$R_squared, f_func)
f_squared<-do.call(rbind,f)
#appending f_squared to data frame
eval_b_df$f_squared<-f_squared


#Generating a 3D graph between log bod mass, slope, and f^2 for the Bayesian optimal escape model
#Putting f^2 into a matrix
f_Squared<-matrix(eval_b_df$f_squared,nrow = 70, ncol=50)

#Creating titles for the 3D graph
axx <- list(
  title = "Log Body Mass (g)"
)

axy <- list(
  title = "Slope"
)

axz <- list(
  title = "f squared"
)


#Plotting the three different variables
surf_b<-plot_ly(x=~mass,y=~slope, z = ~f_Squared, type = "surface", showlegend=T)
#surf_b<- surf_b %>% layout(legend=list(title=list(text='<b>Probability<b>')))
surf_b <- surf_b %>% layout(scene = list(xaxis=axx,yaxis=axy,zaxis=axz))
#surf_b <- surf_b %>% layout(title= "Bayesian Escape Model")
surf_b <- surf_b %>%layout(radialaxis = list(ticksuffix = "%"), orientation = 270)

surf_b
```