General guide to measure contagious yawning and to obtain contagious yawning curves

Please follow the next steps to measure contagious yawning and to obtain contagious yawning curves. For this purpose, we will use a data set from the unfamiliar rats exposed to olfactory communication (download attached files: OC\_Unfamiliar.txt; YCgionUnfRats.txt) and their yawning recorded as integers. See R-code File 1 for more details. To run the program and R commands, we used a Mac OS X platform; some changes might be necessary if a PC platform is used instead.

First, download the data

> my.data <- read.table("Insert here the path to the location of the file OC\_Unfamiliar.txt", header=T) # the path to where the file is located.

> my.data <- data.frame(my.data)

> attach(my.data)

> names(my.data)

[1] "ufr1.l" "ufr2.l" "ufr3.l" "ufr4.l" "ufr5.l" "ufr6.l" "ufr7.l" "ufr8.l" "ufr9.l"

[10] "ufr1.r" "ufr2.r" "ufr3.r" "ufr4.r" "ufr5.r" "ufr6.r" "ufr7.r" "ufr8.r" "ufr9.r"

Then, measure contagious yawning from the perspective of one of the rats and export the data to Excel.

> setwd("Insert here the path to the working directory") # the path to the working directory.

> source("yawn.discrete.R") # commands are previously saved in the working directory; copy them from the R-code File 1. Alternatively, you can copy and paste the commands in the R console.

> pair1.rl <- sapply(1:10, yd, x=ufr1.r,y=ufr1.l) # does the right side rat yawn in response to the left side rat's yawn? Time windows from min 1 to min 10.

[1] 0.1851852 0.1212121

[1] 0.200 0.125

[1] 0.07142857 0.17391304

[1] 0.09090909 0.16326531

[1] 0.0000000 0.1764706

[1] 0.0000000 0.1698113

[1] 0.0000000 0.1698113

[1] 0.0000000 0.1698113

[1] 0.0000000 0.1698113

[1] 0.0000000 0.1698113

> pair1.rl <- t(pair1.rl) # transposition of the matrix

> dimnames(pair1.rl) <- list(c("1","2","3","4","5","6","7","8","9","10"),c("non contagion", "contagion"))# names to rows and columns of the matrix

> pair1.rl # to see the final table

non contagion contagion

1 0.18518519 0.1212121

2 0.20000000 0.1250000

3 0.07142857 0.1739130

4 0.09090909 0.1632653

5 0.00000000 0.1764706

6 0.00000000 0.1698113

7 0.00000000 0.1698113

8 0.00000000 0.1698113

9 0.00000000 0.1698113

10 0.00000000 0.1698113

> write.table(pair1.rl, file="pair1\_rl.txt", sep="\t", col.names=TRUE, row.names=TRUE) # the data are exported to be read from Excel. You can also directly export the data after transposing the matrix.

After that, measure contagious yawning from the perspective of the other rat, and export the data to Excel.

> pair1.lr <- sapply(1:10, yd, x=ufr1.l,y=ufr1.r)# continue as above, and export the data to Excel.

Next, measure contagious yawning again but use a random distribution of the number of yawns of each rat. To generate the distributions, proceed as follows:

> rufr1.r <- sample(ufr1.r,60)

> rufr1.l <- sample(ufr1.l,60)

Note. Sometimes the program displays NaN (Not a Number) instead of a yawn rate value for some time windows. This usually occurs with the “non contagion” computation of the randomly generated data. Such NaNs result because the program finds 0s in the denominator when calculating yawn rates, which is mathematically impossible to perform. In such cases, you can either remove the time windows in which the NaNs appeared from the analyses or substitute them with 0s. The latter option, which is the one we have adopted here, is conservative because one retains the “contagion” rate (*i*.*e*., the occurrence of contagious yawning) in the random data (*i*.*e*., the control group) against which we compare the observed data.

If you recorded yawning to a greater precision (*e*.*g*., decimals of a min), proceed in the same way as described above, but add a fourth argument: the length of the observation period in min (*e*.*g*., 60). See R-code File 2 for more details.

To create the contagion curves, a summary of the descriptive statistics is needed. For this purpose, we will use our dataset from the unfamiliar rats with the 4 test situations.

> dts <- read.table("Insert here the path to the location of the file YCgionUnfRats.txt", header=T) # This is the file where yawn contagion rates were saved.

# First, it is necessary to obtain the difference between contagious and noncontagious yawning for the observed and randomly generated data.

> data <- within(dts, {

ob.yr <- obs.yawn-obs.non.yawn

rnd.yr <- ran.yawn-ran.non.yawn

time.factor <- as.factor(dts$t.window)

})

> data <- data.frame(data)

> attach(data)

# Next, the mean values of the differences are obtained

> ob.bar <- tapply(ob.yr,list(time.factor,exp.con),mean, na.rm=T)

> rnd.bar <- tapply(rnd.yr,list(time.factor,exp.con),mean, na.rm=T)

# Afterwards, the confidence limits are calculated using a bootstrap procedure for the observed data of each test situation first. For this purpose use the next program

> ci <- function(a,b){

dif <- numeric(10000)

for(i in 1:10000){

dif[i] <- mean(sample(a-b, replace=T))}

ci.95 <- quantile(dif,c(0.025,0.975))

print(ci.95)

}

# Before running the program, copying the data in which NAs are removed is required. Such NAs are intentionally used to fill in cells where no value exists (e.g., missing values) or to make all the column lengths equal.

> data2 <- na.omit(data)

> attach(data2)

# Now we can run the program without any trouble

> ci(obs.yawn[exp.con=="oc"&t.window==1],obs.non.yawn[exp.con=="oc"&t.window==1])

2.5% 97.5%

0.01008295 0.141904539

# As this is a random sample, your results might be a little different from the ones obtained here.

# Apply the program for the rest of the time windows and test situations. At the end, you should have something similar to the next numbers for each test situation.

> oci.l <- c(0.01008295,0.01272018,-0.02668742,0.04478325,0.06482535,0.08882158,0.0915661,0.1258690,0.1112676,0.1435705) # lower limit

> oci.u <- c(0.14198164,0.14846036,0.11608537,0.17469681,0.22896778,0.27634176,0.2868937,0.3030017,0.3035634,0.3203746) # upper limit

> vci.l <- c(-0.005214052,-0.01159435,-0.02085587,-0.01698152,-0.02718753,-0.00864097,-0.008130683,-0.002860289,-0.006322891,-0.01926832)

> vci.u <- c(0.097979635,0.08863617,0.09738271,0.12163913,0.12137661,0.15569674,0.167345293,0.172479583,0.176489125,0.17451537)

> nvoci.l <- c(-0.05609855,-0.04383184,-0.03355077,-0.004470844,-0.009933115,-0.005194004,0.006016713,-0.005863838,0.01864638,0.01357083)

> nvoci.u <- c(0.13948770,0.12588185,0.14114744,0.183171196,0.208692872,0.194399248,0.198881102,0.191451379,0.23751849,0.23065252)

> voci.l <- c(-0.10486974,-0.05866717,-0.09507348,-0.11025173,-0.12236149,-0.09009034,-0.06518377,-0.07441788,-0.08708445,-0.07857812)

> voci.u <- c(0.07033302,0.10814355,0.09589300,0.08817397,0.08737716,0.12440182,0.12665336,0.12679259,0.12419907,0.13591615)

# Now, the same program is applied for the randomly generated data.

> ci(ran.yawn[t.window==1],ran.non.yawn[t.window==1])

2.5% 97.5%

-0.04630919 0.02744187

# Do the same for the rest of the time windows to obtain something similar to the following data sets.

> ran.lci <- c(-0.04630919,-0.03887240,-0.04644868,-0.04340323,-0.03449034,-0.05646303,-0.05786448,-0.06967004,-0.07848956,-0.07964649) # lower limit

> ran.uci <- c(0.02744187,0.02870146,0.01778803,0.03086629,0.03969135,0.02245282,0.02577754,0.02271931,0.02903068,0.03367693) # upper limit

# Finally, the graph is produced using all this information

> xv <- seq(1,10)

> par(mfrow=c(2,2),mar=c(4,4,2,2),oma=c(1,1,1,1), lwd=2 ,lend=2, mgp=c(2.5,1,0))

> plot(x="",y="",ylim=c(-0.2,0.4),xlim=c(1,10),xaxp=c(1,10,9),xlab="Time window (min)",ylab="Yawn rate", cex.lab=1.4)

> title(main="A",line=-2,adj=.05, cex.main=1.4) # OC situation

> polygon(c(0:11,11,11:0),c(ran.uci[1],ran.uci,ran.uci[10],ran.lci[10],ran.lci[10],rev(ran.lci),ran.lci[1]),col="lightgrey",border=F)

> abline(h=0,lty=2)

> lines(xv, rnd.bar[,2],lwd=2)

> points(xv,ob.bar[,2],col=2,pch=16,cex=2)

> for(i in 1:10)

lines(c(i,i),c(oci.l[i],oci.u[i]),lwd=2,col=2)

> plot(x="",y="",ylim=c(-0.2,0.4),xlim=c(1,10),xaxp=c(1,10,9),xlab="Time window (min)",ylab="Yawn rate", cex.lab=1.4)

> title(main="B",line=-2,adj=.05, cex.main=1.4) # VC situation

> polygon(c(0:11,11,11:0),c(ran.uci[1],ran.uci,ran.uci[10],ran.lci[10],ran.lci[10],rev(ran.lci),ran.lci[1]),col="lightgrey",border=F)

> abline(h=0,lty=2)

> lines(xv, rnd.bar[,3],lwd=2)

> points(xv,ob.bar[,3],col=4,pch=16,cex=2)

> for(i in 1:10)

lines(c(i,i),c(vci.l[i],vci.u[i]),lwd=2,col=4)

> plot(x="",y="",ylim=c(-0.2,0.4),xlim=c(1,10),xaxp=c(1,10,9),xlab="Time window (min)",ylab="Yawn rate", cex.lab=1.4)

> title(main="C",line=-2,adj=.05, cex.main=1.4)# VOC situation

> polygon(c(0:11,11,11:0),c(ran.uci[1],ran.uci,ran.uci[10],ran.lci[10],ran.lci[10],rev(ran.lci),ran.lci[1]),col="lightgrey",border=F)

> abline(h=0,lty=2)

> lines(xv, rnd.bar[,4],lwd=2)

> points(xv,ob.bar[,4],col=3,pch=16,cex=2)

> for(i in 1:10)

lines(c(i,i),c(voci.l[i],voci.u[i]),lwd=2,col=3)

> plot(x="",y="",ylim=c(-0.2,0.4),xlim=c(1,10),xaxp=c(1,10,9),xlab="Time window (min)",ylab="Yawn rate", cex.lab=1.4)

> title(main="D",line=-2,adj=.05, cex.main=1.4)# NVOC situation

> polygon(c(0:11,11,11:0),c(ran.uci[1],ran.uci,ran.uci[10],ran.lci[10],ran.lci[10],rev(ran.lci),ran.lci[1]),col="lightgrey",border=F)

> abline(h=0,lty=2)

> lines(xv, rnd.bar[,1],lwd=2)

> points(xv,ob.bar[,1],col=6,pch=16,cex=2)

> for(i in 1:10)

lines(c(i,i),c(nvoci.l[i],nvoci.u[i]),lwd=2,col=6)

> dev.off()

# You should see a graph similar to Figure 2 in the main text.