For Fig. 3E: Randomizations

```
##============Parameters=============
##set working directory to source file destination
Nrand <-1000 # how many times do we randomize
modelname <- "model.a.new1" # name of model that is used for output files
modeldef <- expression(latency ~ typeC)

## try to read model from command line
# usage: Rscript randommonkey.R modelname modeldef
args = commandArgs(trailingOnly=TRUE)
if (length(args)==2){
                modelname <- args[1]
                modeldef <- parse(text=args[2])
}

##========================


## load the package glmmADMB, which I used to run the generalized linear models ##
## load instead the package lme4, which works with latest version of RStudio ##
library(lme4)
## load package for dense rank (e.g. (.5,.5,1,2) ranked as (1,1,2,3))
library(dplyr)
#load package for saving histogram
library(ggplot2)

## read in the appropriate file
#file = "File1-data-consensusrank-beisner.csv"
file = "twain6.csv"
data <- read.csv(file, header=T,stringsAsFactors=F)

## fit model to original data ##
##model <- glmmadmb(eval(modeldef), family="nbinom", data=data)
model <- glm(eval(modeldef), family="quasipoisson", data=data)
coefnames <- names(coef(model)[2])

## randomize Nrand times ##
rcoefs <- data.frame(matrix(ncol = length(coefnames), nrow = Nrand))
colnames(rcoefs) <- coefnames

for(ridx in (1:Nrand)){
  write(ridx,stderr())
  rdata <- data

  rdata$latency <- data[sample(1:nrow(data)), 1]
```

```
  rmodel <- glm(eval(modeldef), family="quasipoisson", data=rdata)
  rcoefs[ridx,] = unname(coef(rmodel))[2]
}

### determine p_values for each coefficient
#calculate means
means <- colMeans(rcoefs)
#calculate stddevs
stds <- apply(rcoefs, 2, sd)
#calculate  |random instance - random mean|-|real observation - random mean|
#this is positive if random is further away
#dists <- abs(rcoefs-rep(means,rep.int(nrow(rcoefs), ncol(rcoefs))))-abs(coef(model)-rep(me
#count fraction of times random is further away than real observation, i.e., get the two-sidec
#p_values <- colSums(dists>0)/Nrand

#slow but correct pvalue calculation
p_values <- vector()
# for(coef in names(rcoefs)){
count <- 1.
for(rid in 1:nrow(rcoefs)){
  if(abs(rcoefs[rid,"typeC"]-means)>=abs(coef(model)[2]-means)){
    count = count + 1
  }
}
p_values<-count/nrow(rcoefs)
# }

#Save the Results
summary<-data.frame(as.list(coef(model)))
names(summary)<-names(coef(model))
summary<-cbind(summary,p_values)
summary<-cbind(summary,means)
summary<-cbind(summary,stds)
colnames(summary) <- c('ObsIntercept', 'Obs_typeC', 'p_value','RandomizedMean','Random
write.csv(summary,paste('summary.',modelname,'.rcoefs.csv', sep=''), row.names = T)

#write the sample of randomized coefficients
rownames(rcoefs)<-c(1:Nrand)
write.csv(rcoefs,paste(modelname,'.rcoefs.csv', sep=''), row.names = T)


#save histograms
                myxlim <- c(min(rcoefs[[coefnames]],coef(model)[[coefnames]]),max(rcoefs[[
                myxlim = myxlim +c(-.1,.1)*(myxlim[2]-myxlim[1])
```

```
hist(rcoefs[[coefnames]], col="lightblue", xlab="Regression coefficient", main=
abline(v = -0.55, col="red", lwd=3, lty=2)
```

eans,rep.int(nrow(rcoefs), ncol(rcoefs))))
d p-value

izedStdDev')

coefnames]],coef(model)[[coefnames]]))

'', xlim=c(-1.3,1.3))