```
################################################
### muehlhoff_rcode.R
### Created by Jeffrey R. Stevens on 12 Aug 2010 (jeffrey.r.stevens@gmail.com),
###     finalized on 1 Apr 2011
### Summary: This script calculates descriptive statistics and generates figures
###     for the analysis of spatial discounting in guppies:
###     Muehlhoff, Stevens, & Reader (2011). Spatial discounting of food and
###     social rewards in guppies (Poecilia reticulata). Frontiers in Psychology, 2, 68.
### Instructions: Place the data files (muehlhoff.etal.2011.FiCP_[eval/disc/travel/visual]_data.csv)
###     in a folder labeled 'data' in the same directory as this file.  Create a 'figures' folder
###     in this directory.  Set the R working directory to this directory.
###     At the R command prompt, type
###     > source("muehlhoff.etal.2011.FiCP_rcode.R")
###     This will run the script, adding all of the calculated variables to the
###     workspace and saving PDF versions of the figures in the figures directory.
### Uses: This script can be reproduced and modified for personal and scientific use.
### Data files: Description of the data columns:
###     muehlhoff_eval_data
###      subject - name of subject
###      amount - number of food items and social partners (either 2 or 6)
###      food - number of choices for food
###      social - number of choices for social partners
###     muehlhoff_disc_data
###      date - date of session
###      time - time of session
###      subject - name of subject
###      condition - reward type (f=food and s=social)
###      smallamt - small reward amount (always 2)
###      largeamt - large reward amount (always 6)
###      shortdist - distance to small reward (always 20 cm)
###      longdist - distance to large reward (in cm)
###      forced.free - flag for trial type (either forced or free)
###      trial - trial number within a session
###      left - reward amount on left side
###      right - reward amount on right side
###      choice - side of choice
###      choice.ll - choice for large reward (0=no and 1=yes)
###      initial.main - flag for phase of experiment (i=initial or habituation and m=main experiment)
###      both.conditions - flag for whether subject experience both reward type conditions
###     muehlhoff_travel_data
###      subject - name of subject
###      condition - reward type (f=food and s=social)
###      distance - distance to large reward (in cm)
###      traveltime1 - first measurement of travel time for that subject in that condition and distance
###      traveltime2 - second measurement of travel time for that subject in that condition and distance
###      mean_traveltime - mean travel time for that subject in that condition and distance
###      cond_first - flag for whether this condition was experienced first or second (0=second, 1=first)
###     muehlhoff_visual_data
###      subject - name of subject
###      rewardtype - reward type (f=food and s=social)
###      smallamt - small reward amount (either 0 or 2)
```

```
###      largeamt - large reward amount (6)
###      shortdist - distance to small reward (either 20 or 120 cm)
###      longdist - distance to large reward (120 cm)
###      choicess - number of choices for small reward
###      choicell - number of choices for large reward
#################################################

#################################################
### Load libraries, data, and R version
#################################################
library(epicalc)                                                    # required for aggregating with multiple functions
library(Hmisc)                                                      # required for xYplot
allevaldata <- read.csv("data/muehlhoff.etal.2011.FiCP_eval_data.csv")                # import evaluation data
alldiscdata <- read.csv("data/muehlhoff.etal.2011.FiCP_disc_data.csv")                # import discounting data
travel <- read.csv("data/muehlhoff.etal.2011.FiCP_travel_data.csv")                   # import travel time data
visual <- read.csv("data/muehlhoff.etal.2011.FiCP_visual_data.csv")                   # import visual control data
ver <- getRversion()                                                # get R version

#################################################
## Fish weights
#################################################
weights <- c(0.51, 0.58, 0.55, 0.61, 0.56)                          # measured fish weights (in g)
weights.m <- mean(weights)                                          # calculate mean weight
weights.sd <- sd(weights)                                           # calculate standard deviation of weight

#################################################
## Evaluation phase
#################################################
excludedata <- subset(allevaldata, (amount == -99), drop = T)                         # find rows with missing data
excludesubj <- unique(excludedata$subject)                                            # find subjects with missing data
allsubj <- unique(allevaldata$subj)                                                   # find all subject numbers
includesubj <- allsubj[!allsubj %in% excludesubj]                                     # find all subjects without missing data
evaldata <- subset(allevaldata, subject %in% includesubj)                             # select subjects without missing data
evaldata$pfood <- 100 * evaldata$food / (evaldata$food + evaldata$social)             # calculate the percentage of choices for food
eval2data <- subset(evaldata, amount == 2)                                            # select sessions with amount = 2
eval6data <- subset(evaldata, amount == 6)                                            # select sessions with amount = 6
eval2subj <- aggregate(eval2data$pfood, by = list(eval2data$subject), FUN = mean)     # aggregate over subject
names(eval2subj) <- c("subject", "pfood")                                             # rename columns
eval6subj <- aggregate(eval6data$pfood, by = list(eval6data$subject), FUN = mean)     # aggregate over subject
names(eval6subj) <- c("subject", "pfood")                                             # rename columns
eval2food <- mean(eval2subj$pfood)                                                    # calculate mean choice for food
eval2foodsd <- sd(eval2subj$pfood)                                                    # calculate sd for choice for food
eval2foodN <- length(eval2subj$pfood)                                                 # calculate N for choice for food
eval2foodci <- qt(0.975, df = eval2foodN - 1) * eval2foodsd / sqrt(eval2foodN)        # calculate 95% CI for choice for food
eval6food <- mean(eval6subj$pfood)                                                    # calculate mean choice for food
eval6foodsd <- sd(eval6subj$pfood)                                                    # calculate sd for choice for food
eval6foodN <- length(eval6subj$pfood)                                                 # calculate N for choice for food
eval6foodci <- qt(0.975, df = eval6foodN - 1) * eval6foodsd / sqrt(eval6foodN)        # calculate 95% CI for choice for food


#################################################
```

```r
## Discounting task
##################################################
####################
## All subjects
####################
## Mean number of 20 cm sessions
sessions <- subset(alldiscdata, trial == 1)                                    # select first trials
session20 <- subset(sessions, longdist == 20)                                  # select distance 20 cm trials
session20 <- session20[, c(3, 4, 8)]                                           # use only subject, condition, and longdist columns
session20.agg <- aggregate(session20$longdist,
  by = list(session20$subject, session20$condition), FUN = length)             # aggregate longdist by subject and condition
session20.m <- mean(session20.agg$length)                                      #calculate mean number of sessions at distance 20
session20.sd <- sd(session20.agg$length)                                       #calculate standard deviation of number of sessions at
distance 20

## Discounting aggregated over subjects
mainfree <- subset(alldiscdata, (forced.free == "free" & initial.main == "m"))          # select free choices in the main experiment
allchoice <- aggregate(mainfree$choice.ll, by = list(dist = mainfree$longdist, cond = mainfree$condition),
  FUN = c("mean", "sd", "count"), na.rm = TRUE)                                # aggregate choice by distance and condition
names(allchoice) <- c("distance", "condition", "pll", "sd", "N")               # rename columns
allchoice$pll <- 100 * allchoice$pll                                           # convert proportions to percentages
allchoice$sd <- 100 * allchoice$sd                                             # convert proportions to percentages
allchoice$ci <- qt(0.975, df = allchoice$N - 1) * allchoice$sd / sqrt(allchoice$N)      # calculate 95% CI for choice for larger

# Descriptive statistics
f20 <- allchoice[allchoice$distance == 20 & allchoice$condition == "f", ]      # select food, 20 cm data
s20 <- allchoice[allchoice$distance == 20 & allchoice$condition == "s", ]      # select social, 20 cm data
f120 <- allchoice[allchoice$distance == 120 & allchoice$condition == "f", ]    # select food, 120 cm data
s120 <- allchoice[allchoice$distance == 120 & allchoice$condition == "s", ]    # select social, 120 cm data
all20 <- mean(f20$pll, s20$pll)                                                # calculate mean at 20 cm

food.soc.col <- c("#0072B2","#E69F00")

# Plot discounting aggregated over subjects
pdf(file = "figures/muehlhoff_fig3.pdf", width = 10)
allchoice_plot <- xYplot(Cbind(pll, pll + ci, pll - ci) ~ distance, groups = condition, data = allchoice, type = "b",
  ylim = c(-5, 105), xlab = "Distance to larger reward (cm)", ylab = "Percent choosing larger reward\n(mean ± 95% CI)",
  aspect = 0.75, cex = 1.5, col = food.soc.col, lwd = 3.5, label.curves = FALSE, pch = c(19, 17),
  par.settings = list(axis.text = list(cex = 1.75), par.xlab.text = list(cex = 2), par.ylab.text = list(cex = 2)),
  key = list(corner = c(0.95, 0.95), padding.text = 3, cex = 1.75,
    text = list(c("Food", "Social"), adj = 1),
    lines = list(col = food.soc.col), lwd = 3.5, lty = c(1, 1), pch = c(19, 17), type = c("b", "b"), divide = 1)
)
plot(allchoice_plot)
dev.off()

# Regression equation
allfood <- subset(allchoice, condition == "f")                                 # select food data
allsocial <- subset(allchoice, condition == "s")                               # select social data
allfood_coef <- coef(lm(pll ~ distance, data = allfood))                       # calculate linear regression coefficients for food data
allsocial_coef <- coef(lm(pll ~ distance, data = allsocial))                   # calculate linear regression coefficients for social data
```

```
allfood_inter <- allfood_coef[1]                                        # extract food intercept
allfood_slope <- allfood_coef[2]                                        # extract food slope
allsocial_inter <- allsocial_coef[1]                                    # extract social intercept
allsocial_slope <- allsocial_coef[2]                                    # extract social slope
names(allfood_inter) <- NULL
names(allfood_slope) <- NULL
names(allsocial_inter) <- NULL
names(allsocial_slope) <- NULL
allfood_indiff <- (50 - allfood_inter) / allfood_slope                  # calculate distance at indifference for food data
allsocial_indiff <- (50 - allsocial_inter) / allsocial_slope            # calculate distance at indifference for social data

## Discounting at the individual level
subjchoice <- aggregate(mainfree$choice.ll, by = list(mainfree$longdist, mainfree$condition, mainfree$subject, mainfree$both.conditions),
  FUN = c("mean", "count", "sd"), na.rm = TRUE)                                                    # aggregate choice by distance,
condition, subject, and both conditions
names(subjchoice) <- c("distance", "condition", "subject", "within", "pll", "N", "sd")
subjchoice$pll <- 100 * subjchoice$pll                                                           # convert proportions to percentages
subjchoice$sd <- 100 * subjchoice$sd                                                             # convert proportions to percentages
subjchoice$within <- as.factor(ifelse(subjchoice$within == "n", ifelse(subjchoice$cond == "s", "s", "f"), "b")) # generate factor describing
whether subject experienced both conditions
subjchoice$subj <- factor(subjchoice$subject, levels(subjchoice$subject)[c(8, 12, 1, 3, 4, 5, 13, 6, 7, 9, 14, 2, 10, 11)])

# Plot of individual discounting choices
pdf(file = "figures/muehlhoff_fig4.pdf", width = 12)
subjchoice_plot <- xYplot(pll ~ distance | subj, groups = condition, data = subjchoice, type = "b",
  label.curves = FALSE, col = food.soc.col, pch = c(19, 17), lwd = 3.5, layout = c(5, 3), cex = 1.25,
  ylim = c(-5, 105), xlab = "Distance to larger reward (cm)", ylab = "Percent choosing larger reward",
  aspect = 0.75, as.table = TRUE, strip = strip.custom(strip.names = FALSE, par.strip.text = list(cex = 1.5)),
  par.settings = list(axis.text = list(cex = 1.5), par.xlab.text = list(cex = 2), par.ylab.text = list(cex = 2),
  layout.heights = list(strip = 1.5)),
  key = list(corner = c(0.99, 0.03), padding.text = 3, cex = 1.5,
      text = list(c("Food", "Social"), adj = 1),
      lines = list(col = food.soc.col), lwd = 3.5, lty = c(1, 1), pch = c(19, 17), type = c("b", "b"), divide = 1)
)
plot(subjchoice_plot)
dev.off()

## Retinal area of food markers
w <- 2                                  # width
h <- 1                                  # height
d <- 20:120                             # distances
Rw <- tan(2 * atan(w / (2 * d)))        # relative retinal width (assuming eye is 1 unit in diameter)
Rh <- tan(2 * atan(h / (2 * d)))        # relative retinal height
Ra <- Rw * Rh                           # relative retinal area for one strip
Ra_approx <- (w*h)/(d ^ 2)              # approximate retinal area for one strip
Ral <- 6 * Ra                           # relative retinal area for six strips across all distances
Rac <- rep(2 * Ra[1], length(d))        # relative retinal area for two strips at distance 20cm

Rarea <- data.frame(d, Rac, Ral)
names(Rarea) <- c("distance", "smallarea", "largearea")
Rarea$diff <- Rarea$largearea - Rarea$smallarea         # find difference between areas
```

```
prefsmall <- subset(Rarea, largearea - smallarea < 0)   # extract distances with larger area for smaller reward
switchpt <- min(prefsmall$dist)                         # find switchpoint (minimum distance with larger area for smaller reward"


####################
## Within subjects
####################
## Subjects with both conditions
within <- subset(mainfree, both.conditions == "y")                      # select data from subjects that completed both conditions
wsubj.choice <- aggregate(within$choice.ll, by = list(within$longdist, within$condition, within$subject),
  FUN = c("mean", "count"), na.rm = TRUE)                               # aggregate choice over distance, condition, and subject
names(wsubj.choice) <- c("dist", "cond", "subj", "pll", "Nll")
wsubj.choice$tpll <- asin(sqrt(wsubj.choice$pll))                       # create arcsine, square-root transformed proportion of choices for larger,
farther
wsubj.choice$fdist <- as.factor(wsubj.choice$dist)                      # convert distance to a factor


## Tests of ANOVA assumptions
#Normality
raw.norm <- shapiro.test(wsubj.choice$pll)              # calculate Shapiro's test for normality on raw data
shapiro.W <- raw.norm$statistic                         # extract test statistic
names(shapiro.W) <- NULL
shapiro.p <- raw.norm$p.value                           # extract p value
trans.norm <- shapiro.test(wsubj.choice$tpll)           # calculate Shapiro's test for normality on transformed data
tshapiro.W <- trans.norm$statistic                      # extract test statistic
names(tshapiro.W) <- NULL
tshapiro.p <- trans.norm$p.value                        # extract p value

# Homogeneity of variance
wsubj.choice$levels <- paste(wsubj.choice$fdist, wsubj.choice$cond, sep = "")   # create vector of all levels
Levene <- function(y, group) {                                                  # Levene's test of homogeneity of variance (by Brian Ripley)
     group <- as.factor(group)  # precautionary
     meds <- tapply(y, group, median)
     resp <- abs(y - meds[group])
     anova(lm(resp ~ group))[1, 4:5]
 }
raw.levene <- Levene(wsubj.choice$pll, wsubj.choice$levels)     # calculate Levene's test on raw data
levene.F <- raw.levene$"F value"                                # extract F value
levene.p <- raw.levene$"Pr(>F)"                                 # extract p value
trans.levene <- Levene(wsubj.choice$tpll, wsubj.choice$levels)  # calculate Levene's test on raw data
tlevene.F <- trans.levene$"F value"                             # extract F value
tlevene.p <- trans.levene$"Pr(>F)"                              # extract p value


## ANOVA
options(contrasts=c("contr.sum","contr.poly"))
rmaov <- aov(pll ~ fdist * cond + Error(subj / (fdist * cond)), data = wsubj.choice)    # Repeated measures ANOVA
trmaov <- aov(tpll ~ fdist * cond + Error(subj / (fdist * cond)), data = wsubj.choice)  # Repeated measures ANOVA with transformed data
sumaov <- summary(trmaov)
subjss <- sumaov$"Error: subj"[[1]]$"Sum Sq"[1]                                         # extract subject sums of squares (s in Bakeman 2005)
distdf1 <- sumaov$"Error: subj:fdist"[[1]]$"Df"[1]                                      # extract distance degrees of freedom (numerator)
```

```r
distdf2 <- sumaov$"Error: subj:fdist"[[1]]$"Df"[2]                                                    # extract distance degrees of freedom (denomenator)
distF <- sumaov$"Error: subj:fdist"[[1]]$"F value"[1]                                                 # extract distance F value
distp <- sumaov$"Error: subj:fdist"[[1]]$"Pr(>F)"[1]                                                  # extract distance p value
distss <- sumaov$"Error: subj:fdist"[[1]]$"Sum Sq"[1]                                                 # extract distance sums of squares (P in Bakeman 2005)
distsserror <- sumaov$"Error: subj:fdist"[[1]]$"Sum Sq"[2]                                            # extract distance error sums of squares (Ps in Bakeman
2005)
conddf1 <- sumaov$"Error: subj:cond"[[1]]$"Df"[1]                                                     # extract condition degrees of freedom (numerator)
conddf2 <- sumaov$"Error: subj:cond"[[1]]$"Df"[2]                                                     # extract condition degrees of freedom (numerator)
condF <- sumaov$"Error: subj:cond"[[1]]$"F value"[1]                                                  # extract condition F value
condp <- sumaov$"Error: subj:cond"[[1]]$"Pr(>F)"[1]                                                   # extract condition F value
condss <- sumaov$"Error: subj:cond"[[1]]$"Sum Sq"[1]                                                  # extract condition sums of squares (Q in Bakeman 2005)
condsserror <- sumaov$"Error: subj:cond"[[1]]$"Sum Sq"[2]                                             # extract condition sums of squares (Qs in Bakeman 2005)
distconddf1 <- sumaov$"Error: subj:fdist:cond"[[1]]$"Df"[1]                                           # extract distance*condition degrees of freedom (numerator)
distconddf2 <- sumaov$"Error: subj:fdist:cond"[[1]]$"Df"[2]                                           # extract distance*condition degrees of freedom (numerator)
distcondF <- sumaov$"Error: subj:fdist:cond"[[1]]$"F value"[1]                                        # extract distance*condition F value
distcondp <- sumaov$"Error: subj:fdist:cond"[[1]]$"Pr(>F)"[1]                                         # extract distance*condition F value
distcondss <- sumaov$"Error: subj:fdist:cond"[[1]]$"Sum Sq"[1]                                        # extract distance*condition sums of squares (PQ in Bakeman
2005)
distcondsserror <- sumaov$"Error: subj:fdist:cond"[[1]]$"Sum Sq"[2]                                   # extract distance*condition sums of squares (PQs in
Bakeman 2005)
disteta <- distss / (distss + distsserror)                                                            # calculate distance partial eta squared
distgeta <- distss / (distss + subjss + distsserror + condsserror + distcondsserror)                  # calculate distance generalized eta squared
condeta <- condss / (condss + condsserror)                                                            # calculate condition partial eta squared
condgeta <- condss / (condss + subjss + distsserror + condsserror + distcondsserror)                  # calculate condition generalized eta squared
distcondeta <- distcondss / (distcondss + distcondsserror)                                            # calculate distance*condition partial eta squared
distcondgeta <- distcondss / (distcondss + subjss + distsserror + condsserror + distcondsserror)      # calculate distance*condition generalized
eta squared

#####################
## Travel time
#####################
names(travel) <- c("subject", "condition", "distance", "traveltime1", "traveltime2", "time", "speed")
travel$fdist <- as.factor(travel$dist)                                                                # convert distance to factor
choicetravel <- merge(subjchoice, travel)                                                             # merge choice and travel data
withintravel <- subset(choicetravel, within == "b")                                                   # select subjects in both conditions

# Plot of travel time as a function of distance
choicetravel$condition <- as.factor(ifelse(choicetravel$condition == "f", "Food",
  ifelse(choicetravel$condition == "s", "Social", "NA")))                                             # recode condition as Food, Social, and NA
pdf(file = "figures/muehlhoff_fig5.pdf", width = 11, height = 6)
time_bwplot <- bwplot(time ~ condition | fdist, data = choicetravel,
  xlab = "Reward type", ylab = "Travel time (s)", col = food.soc.col, layout = c(6,1), #aspect = 0.75,
  strip = strip.custom(factor.levels = c("20 cm", "40 cm", "60 cm", "80 cm", "100 cm", "120 cm"), par.strip.text = list(cex = 1.5)),
  par.settings = list(axis.text = list(cex = 1.5), par.xlab.text = list(cex = 2), par.ylab.text = list(cex = 2),
    layout.heights = list(strip = 1.5), box.umbrella = list(lty = 1, col = "black", lwd = 2),
    box.rectangle = list(lwd = 2, col = "black"), fill = food.soc.col),
  panel = function(x, y, groups) {
    panel.bwplot(x, y, pch = "|", horizontal = F, coef = 0, fill = food.soc.col)
    mean.values <<- tapply(y, x, mean, na.rm=T)                          #calculates means
    panel.points(mean.values, pch = 18, cex = 1.5, col = "black")       #plots means as diamonds
  }
```

```
)
plot(time_bwplot)
dev.off()

foodtravel120 <- subset(choicetravel, (distance == 120 & condition == "Food"))          # select food trials at 120 cm
socialtravel120 <- subset(choicetravel, (distance == 120 & condition == "Social"))      # select social trials at 120 cm
foodsocialratio <- mean(foodtravel120$time) / mean(socialtravel120$time) * 100          # calculate ratio of mean travel times in food and social
conditions

# Plot of choice for larger reward as a function of travel time
pdf(file = "figures/muehlhoff_fig6.pdf", width = 8, height = 6)
timedisc_plot <- xyplot(pll ~ time, data = choicetravel, groups = condition, type=c("p","r"),
  xlab = "Travel time (s)", ylab = "Percent choosing larger reward", ylim = c(-5, 105),
  col = food.soc.col, lwd = 2, pch = c(19, 17), cex = 1.5, aspect = 0.75,
  par.settings = list(axis.text = list(cex = 1.25), par.xlab.text = list(cex = 1.75), par.ylab.text = list(cex = 1.75)),
  key = list(corner = c(0.95, 0.95), padding.text = 3, cex = 1.5,
    text = list(c("Food", "Social"), adj = 1),
    lines = list(col = food.soc.col), lwd = 2, lty = c(1, 1), pch = c(19, 17), type = c("b", "b"), divide = 1),
  panel = function(...){
    panel.xyplot(...)
    panel.abline(h = 50, lty = 2, lwd = 2)
  }
)
plot(timedisc_plot)
dev.off()

#Regression lines
foodtime <- subset(choicetravel, condition == "Food")           # select food travel data
socialtime <- subset(choicetravel, condition == "Social")       # select social travel data
fooddisc_coef <- coef(lm(pll ~ time, data = foodtime))          # calculate linear regression coeficients for food data
socialdisc_coef <- coef(lm(pll ~ time, data = socialtime))      # calculate linear regression coeficients for social data
fooddisc_inter <- fooddisc_coef[1]                              # extract food intercept
fooddisc_slope <- fooddisc_coef[2]                              # extract food slope
socialdisc_inter <- socialdisc_coef[1]                          # extract social intercept
socialdisc_slope <- socialdisc_coef[2]                          # extract social slope
names(fooddisc_inter) <- NULL
names(fooddisc_slope) <- NULL
names(socialdisc_inter) <- NULL
names(socialdisc_slope) <- NULL
fooddisc_indiff <- (50 - fooddisc_inter) / fooddisc_slope       # calculate distance at indifference for food data
socialdisc_indiff <- (50 - socialdisc_inter) / socialdisc_slope # calculate distance at indifference for social data


################################################
## Visual control
################################################
visual$pll <- 100 * visual$choicell / (visual$choicell + visual$choicess)              # calculate percent choices for LL option in visual control
task

## Forced visual control task
# All data
```

```r
forced <- subset(visual, smallamt == 0)                                                    # select data for forced visual control task
forcedsubj <- aggregate(forced$pll, by = list(forced$subject), FUN = mean)                 # aggregate choice data by subject
names(forcedsubj) <- c("subject", "pll")
forcedsubj.m <- mean(forcedsubj$pll)                                                       # calculate mean choice for large reward
forcedsubj.sd <- sd(forcedsubj$pll)                                                        # calculate standard deviation choice for large reward
forcedsubj.N <- length(forcedsubj$subject)                                                 # calculate N
forcedsubj.ci <- qt(0.975, df = forcedsubj.N - 1) * forcedsubj.sd / sqrt(forcedsubj.N)     # calculate 95% CI for choice for large reward
forcedsubj.min <- min(forced$pll)                                                          # calculate minimum choice for large reward
forcedsubj.max <- max(forced$pll)                                                          # calculate maximum choice for large reward
forcedreward <- aggregate(forced$pll, by = list(forced$subject, forced$rewardtype), FUN = mean) # aggregate choice data by subject and reward type
names(forcedreward) <- c("subject", "rewardtype", "pll")
# Food data
forcedfood <- subset(forcedreward, rewardtype == "f")                                      # select food data (by subject and reward type)
forcedfood.m <- mean(forcedfood$pll)                                                       # calculate mean choice for large reward
forcedfood.sd <- sd(forcedfood$pll)                                                        # calculate sd choice for large reward
forcedfood.N <- length(forcedfood$subject)                                                 # calculate N
forcedfood.ci <- qt(0.975, df = forcedfood.N - 1) * forcedfood.sd / sqrt(forcedfood.N)     # calculate 95% CI for choice for larger
forcedfood.min <- min(forcedfood$pll)                                                      # calculate minimum choice for large reward
forcedfood.max <- max(forcedfood$pll)                                                      # calculate maximum choice for large reward
# Social data
forcedsocial <- subset(forcedreward, rewardtype == "s")                                    # select social data (by subject and reward type)
forcedsocial.m <- mean(forcedsocial$pll)                                                   # calculate mean choice for large reward
forcedsocial.sd <- sd(forcedsocial$pll)                                                    # calculate sd choice for large reward
forcedsocial.N <- length(forcedsocial$subject)                                             # calculate N
forcedsocial.ci <- qt(0.975, df = forcedsocial.N - 1) * forcedsocial.sd / sqrt(forcedsocial.N)  # calculate 95% CI for choice for larger
forcedsocial.min <- min(forcedsocial$pll)                                                  # calculate minimum choice for large reward
forcedsocial.max <- max(forcedsocial$pll)                                                  # calculate maximum choice for large reward
forcedfood.t <- t.test(pll ~ rewardtype, data = forcedreward, paired = T)                  # calculate t-test comparing reward type

## Split visual control task
# All data
split <- subset(visual, smallamt == 2)                                                     # select split visual control data
splitsubj <- aggregate(split$pll, by = list(split$subject), FUN = mean)                    # aggregate choice data by subject
names(splitsubj) <- c("subject", "pll")
splitsubj.m <- mean(splitsubj$pll)                                                         # calculate mean choice for large reward
splitsubj.sd <- sd(splitsubj$pll)                                                          # calculate sd choice for large reward
splitsubj.N <- length(splitsubj$subject)                                                   # calculate N
splitsubj.ci <- qt(0.975, df = splitsubj.N - 1) * splitsubj.sd / sqrt(splitsubj.N)         # calculate 95% CI for choice for larger
splitsubj.min <- min(split$pll)                                                            # calculate minimum choice for large reward
splitsubj.max <- max(split$pll)                                                            # calculate maximum choice for large reward
splitfoodall <- subset(split, rewardtype == "f")                                           # select food data (by subject)
splitsocialall <- subset(split, rewardtype == "s")                                         # select social data (by subject)
splitreward <- aggregate(split$pll, by = list(split$subject, split$rewardtype), FUN = mean)     # aggregate choice by subject and reward type
names(splitreward) <- c("subject", "rewardtype", "pll")
# Food data
splitfood <- subset(splitreward, rewardtype == "f")                                        # select food data  (by subject and reward type)
splitfood.m <- mean(splitfood$pll)                                                         # calculate mean choice for large reward
splitfood.sd <- sd(splitfood$pll)                                                          # calculate sd choice for large reward
splitfood.N <- length(splitfood$subject)                                                   # calculate N
splitfood.ci <- qt(0.975, df = splitfood.N - 1) * splitfood.sd / sqrt(splitfood.N)         # calculate 95% CI for choice for larger
splitfood.min <- min(splitfood$pll)                                                        # calculate minimum choice for large reward
```

```r
splitfood.max <- max(splitfood$pll)                                                 # calculate maximum choice for large reward
# Social data
splitsocial <- subset(splitreward, rewardtype == "s")                               # select social data  (by subject and reward type)
splitsocial.m <- mean(splitsocial$pll)                                              # calculate mean choice for large reward
splitsocial.sd <- sd(splitsocial$pll)                                               # calculate sd choice for large reward
splitsocial.N <- length(splitsocial$subject)                                        # calculate N
splitsocial.ci <- qt(0.975, df = splitsocial.N - 1) * splitsocial.sd / sqrt(splitsocial.N)      # calculate 95% CI for choice for larger
splitsocial.min <- min(splitsocial$pll)                                             # calculate minimum choice for large reward
splitsocial.max <- max(splitsocial$pll)                                             # calculate maximum choice for large reward
splitfood.t <- t.test(pll ~ rewardtype, data = splitreward, paired = T)             # calculate t-test comparing reward type
```