Liking effect induced by gaze

Necdet Gürkan & Yangyang Yu

Last Knitted: 2023-09-18

### Vignette Setup:

### Project/Data Title:

Liking effect induced by gaze

Data provided by: José Luis Ulloa, Clara Marchetti, Marine Taffou & Nathalie George

### Project/Data Description:

This dataset resulted from a study aiming at investigating how gaze perception can influence preferences. Previous studies suggest that we like more the objects that are looked-at by others than non-looked-at objects (a so-called liking effect). We extended previous studies to investigate both abstract and manipulable objects. Participants performed a categorization task (for items that were cued or not by gaze). Next, participants evaluated how much they liked the items. We tested if the liking effect could be observed for non-manipulable (alphanumeric characters) as well as for manipulable items (common tools).

### Methods Description:

Participants were students at Heinrich-Heine-Universität Düsseldorf, Germany that were paid for participating or received course credit. Their ages ranged from 18 to 55 years. The words to-be-rated consisted of 30 typical members of 30 categories drawn from the updated Battig and Montague norms (Van Overschelde, Rawson, & Dunlosky, 2004).

### Data Location:

Data included within this vignette.

DF <- import("ulloa\_data.csv")  
drops <- c("RT", "side", "aff-ness")  
DF <- DF[ , !(names(DF) %in% drops)]  
head(DF)

## suj congr item liking  
## 1 1 valid G 4  
## 2 1 valid G 3  
## 3 1 valid G 1  
## 4 1 valid G 3  
## 5 1 valid K 8  
## 6 1 valid K 6

### Date Published:

No official publication, see citation below.

### Dataset Citation:

José Luis Ulloa, Clara Marchetti, Marine Taffou & Nathalie George (2014): Only your eyes tell me what you like: Exploring the liking effect induced by other’s gaze, Cognition & Emotion, DOI: 10.1080/02699931.2014.919899

### Keywords:

Social attention; Gaze; Pointing gesture; Liking; Cueing.

### Use License:

CC-BY

### Geographic Description - City/State/Country of Participants:

Paris, France

### Column Metadata:

metadata <- import("ulloa\_metadata.xlsx")  
  
flextable(metadata) %>% autofit()

| Variable Name | Variable Description | Type (numeric, character, logical, etc.) |
| --- | --- | --- |
| suj | Unique number assigned to each participant | Numeric |
| congr | valid vs invalid | Character |
| item | G, K, S, L | Character |
| liking | rating response | Numeric |

### AIPE Analysis

In this dataset, there are valid and invalid cue-targeting variable. In valid cue-targeting condition, stimulus is on the same side of the gaze. In invalid cue-targeting condition, stimulus was on the opposite side of the gaze. We consider these two different conditions separately.

#### Stopping Rule

What the usual standard error for the data that could be considered for our stopping rule using the 40% decile?

### create subset for valid cue-targeting  
DF\_valid <- subset(DF, congr == "valid") %>%   
 group\_by(suj, item) %>%   
 summarize(liking = mean(liking, na.rm = T)) %>%   
 as.data.frame()

## `summarise()` has grouped output by 'suj'. You can override using the `.groups`  
## argument.

### create subset for invalid cue-targeting  
DF\_invalid <- subset(DF, congr == "invalid") %>%   
 group\_by(suj, item) %>%   
 summarize(liking = mean(liking, na.rm = T)) %>%   
 as.data.frame()

## `summarise()` has grouped output by 'suj'. You can override using the `.groups`  
## argument.

# individual SEs for valid cue-targeting condition   
SE1 <- tapply(DF\_valid$liking, DF\_valid$item, function (x) { sd(x)/sqrt(length(x)) })  
  
SE1

## G K L S   
## 0.2013228 0.1779694 0.1801060 0.2286006

cutoff1 <- quantile(SE1, probs = .4)  
cutoff1

## 40%   
## 0.1843494

# individual SEs for invalid cue-targeting condition  
SE2 <- tapply(DF\_invalid$liking, DF\_invalid$item, function (x) { sd(x)/sqrt(length(x)) })  
  
SE2

## G K L S   
## 0.1982333 0.1749820 0.1724613 0.2132725

cutoff2 <- quantile(SE2, probs = .4)  
cutoff2

## 40%   
## 0.1796323

# sequence of sample sizes to try  
samplesize\_values <- seq(25, 200, 5)  
  
# create a blank table for us to save the values in   
sim\_table <- matrix(NA,   
 nrow = length(samplesize\_values),   
 ncol = length(unique(DF\_valid$item)))  
# make it a data frame  
sim\_table <- as.data.frame(sim\_table)  
  
# add a place for sample size values   
sim\_table$sample\_size <- NA  
sim\_table$var <- "liking"  
  
# make a second table for the second variable  
sim\_table2 <- matrix(NA,   
 nrow = length(samplesize\_values),   
 ncol = length(unique(DF\_valid$item)))  
  
# make it a data frame  
sim\_table2 <- as.data.frame(sim\_table2)  
  
# add a place for sample size values   
sim\_table2$sample\_size <- NA  
sim\_table2$var <- "liking"  
  
# loop over sample sizes for age and outdoor trait  
for (i in 1:length(samplesize\_values)){  
   
 # temp dataframe for age trait that samples and summarizes  
 temp\_valid <- DF\_valid %>%   
 dplyr::group\_by(item) %>%   
 dplyr::sample\_n(samplesize\_values[i], replace = T) %>%   
 dplyr::summarize(se1 = sd(liking)/sqrt(length(liking)))   
   
 #   
 colnames(sim\_table)[1:length(unique(DF\_valid$item))] <- temp\_valid$item  
 sim\_table[i, 1:length(unique(DF\_valid$item))] <- temp\_valid$se1  
 sim\_table[i, "sample\_size"] <- samplesize\_values[i]  
   
 # temp dataframe for outdoor trait that samples and summarizes  
   
 temp\_invalid <-DF\_invalid %>%   
 dplyr::group\_by(item) %>%   
 dplyr::sample\_n(samplesize\_values[i], replace = T) %>%   
 dplyr::summarize(se2 = sd(liking)/sqrt(length(liking)))   
  
 #   
 colnames(sim\_table)[1:length(unique(DF\_invalid$item))] <- temp\_invalid$item  
 sim\_table2[i, 1:length(unique(DF\_invalid$item))] <- temp\_invalid$se2  
 sim\_table2[i, "sample\_size"] <- samplesize\_values[i]  
}

Calculate the cutoff score with information necessary for correction.

cutoff\_valid <- calculate\_cutoff(population = DF\_valid,   
 grouping\_items = "item",  
 score = "liking",   
 minimum = min(DF\_valid$liking),  
 maximum = max(DF\_valid$liking))  
  
# same as above  
cutoff\_valid$cutoff

## 40%   
## 0.1843494

cutoff\_invalid <- calculate\_cutoff(population = DF\_invalid,   
 grouping\_items = "item",  
 score = "liking",   
 minimum = min(DF\_valid$liking),  
 maximum = max(DF\_valid$liking))  
  
cutoff\_invalid$cutoff

## 40%   
## 0.1796323

### for valid cue-targeting condition  
final\_sample\_valid <-   
 sim\_table %>%  
 pivot\_longer(cols = -c(sample\_size, var)) %>%   
 dplyr::rename(item = name, se = value) %>%   
 dplyr::group\_by(sample\_size, var) %>%   
 dplyr::summarize(percent\_below = sum(se <= cutoff1)/length(unique(DF\_valid$item))) %>%   
 dplyr::arrange(percent\_below) %>%   
 ungroup()

## `summarise()` has grouped output by 'sample\_size'. You can override using the  
## `.groups` argument.

flextable(final\_sample\_valid %>% head()) %>%   
 autofit()

| sample\_size | var | percent\_below |
| --- | --- | --- |
| 25 | liking | 0.25 |
| 30 | liking | 0.25 |
| 40 | liking | 0.25 |
| 35 | liking | 0.50 |
| 50 | liking | 0.50 |
| 45 | liking | 0.75 |

Calculate the final corrected scores:

final\_scores <- calculate\_correction(proportion\_summary = final\_sample\_valid,  
 pilot\_sample\_size = length(unique(DF$suj)),  
 proportion\_variability = cutoff\_valid$prop\_var)  
  
# only show first four rows since all 100  
flextable(final\_scores %>%   
 ungroup() %>%   
 slice\_head(n = 4)) %>% autofit()

| percent\_below | sample\_size | corrected\_sample\_size |
| --- | --- | --- |
| 100 | 55 | 50.74449 |
| 100 | 55 | 50.74449 |
| 100 | 55 | 50.74449 |
| 100 | 55 | 50.74449 |

### for valid cue-targeting condition  
final\_sample\_invalid <-   
 sim\_table2 %>%  
 pivot\_longer(cols = -c(sample\_size, var)) %>%   
 dplyr::rename(item = name, se = value) %>%   
 dplyr::group\_by(sample\_size, var) %>%   
 dplyr::summarize(percent\_below = sum(se <= cutoff2)/length(unique(DF\_invalid$item))) %>%   
 dplyr::arrange(percent\_below) %>%   
 ungroup()

## `summarise()` has grouped output by 'sample\_size'. You can override using the  
## `.groups` argument.

flextable(final\_sample\_invalid %>% head()) %>%   
 autofit()

| sample\_size | var | percent\_below |
| --- | --- | --- |
| 25 | liking | 0.00 |
| 30 | liking | 0.50 |
| 35 | liking | 0.50 |
| 45 | liking | 0.50 |
| 40 | liking | 0.75 |
| 50 | liking | 0.75 |

Calculate the final corrected scores:

final\_scores2 <- calculate\_correction(proportion\_summary = final\_sample\_invalid,  
 pilot\_sample\_size = length(unique(DF$suj)),  
 proportion\_variability = cutoff\_invalid$prop\_var)  
  
# only show first four rows since all 100  
flextable(final\_scores2 %>%   
 ungroup() %>%   
 slice\_head(n = 4)) %>% autofit()

| percent\_below | sample\_size | corrected\_sample\_size |
| --- | --- | --- |
| 100 | 55 | 51.09465 |
| 100 | 55 | 51.09465 |
| 100 | 55 | 51.09465 |
| 100 | 55 | 51.09465 |

#### Minimum Sample Size

Based on these simulations, we can decide our minimum sample size for 80% is likely close to 51 for the valid trials or 51 for the invalid trials, depending on rounding.

#### Maximum Sample Size

In this example, we could set our maximum sample size for 95% items below the criterion, which would equate to 51 for the valid trials or 51 for invalid trials. In this case, values are equal because the percent below jumps from 75% to 100%.

#### Final Sample Size

In any estimate for sample size for this study, the dataset has a large variance in ratings. This dataset need to more sample for items in each conditions. In fact, we experimented combining two conditions (valid & invalid cue-targeting) which did not result in any difference.