Survey Two - Image Recognition - Data Cleaning

Hari VS

```r
library(tidyverse) #for tidyr and new tidy version coding
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(dplyr) #for pipe usage
library(lubridate) #to handle date and time
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

RAW DATA OBTAINED FROM PROLIFIC THROUGH SURVEY

THIS FILE CONTAINS ONLY THE CLEANING CODE

DATA IS CLEANED AND SAVED IN MULTIPLE FORMATS

1. EACH PERSON IS AN OBSERVATION

```
    1A. AVERAGED ACROSS ALL IMAGES
    1B. AVERAGED ACROSS PLANTS ONLY IMAGES
    1C. AVERAGED ACROSS ANIMALS ONLY IMAGES
```

```r
# import raw data from Qualtrics
raw_data <- read_csv("Datasets/study_data.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character()
## )
## i Use `spec()` for the full column specifications.
```

```
#copying data so originial is left untouched
data_person <- raw_data

#Removing first two unwanted rows. Duplicates for column names
data_person <- data_person[-c(1,2),]

#Converting class of the timestamps
data_person$StartDate <-  as_datetime(data_person$StartDate)

#started collecting data on 2/11/21 at 6:30 PM
data_person <-
  filter(data_person, StartDate >= #select cells
as.POSIXct("2021-02-11 18:30:00", #that has timestamps past 6:30 PM on 02/11/21
tz = "UTC")) %>% #timestamps are in Mountrain TimeZone
  filter(Finished == "True") #removing incomplete surveys

#Removing unwanted columns
data_person <- select(data_person, -c("EndDate": "Progress","RecordedDate",
                      "RecipientLastName" : "UserLanguage", "Comments",
                      "mTurkCode")) #Specifying column names

# remove unused factor levels
data_person <- droplevels(data_person)

# convert to tibble
data_person <- as_tibble(data_person)

#Removing columns that has NAs for all cells
all_na <- function(x){
  any(!is.na(x)) #anything is that not NA.
#write the names of the columns.
}

#Applying the all_na funtion to data_person tibble
data_person <- data_person[, #chossing columns from the data frame
                           which( #mentioning which columns
                             unlist( #drop the one that follows lapply
                               lapply(data_person,all_na)))]
                               #lapply tells to apply all_na function

#Removing more unwanted columns
data_person <- data_person %>%
  select(-contains("Click")) #These columns are not needed
#These columns recorded the first click and last click time (in seconds) for
#every stimuli. It also recorded click counts for all.

#Removing Hannah from dataset -
#Survey Preview done by peer during data collection
data_person <- data_person[data_person$`P-ID` != "Hannah",]
```

Age

```r
#convert to numeric
data_person$Year_born <- as.numeric(data_person$Year_born)

# calculate age
data_person$age <- 2020 - data_person$Year_born #numeric class
```

Education

```r
# Creating dummy variable for EDUCATION
data_person$college <- #4-year degree or higher = 1, else = 0.
  #case_when: if response is equal to "x", assign "y"
  case_when(
    data_person$Education == "Graduate degree" ~ 1,
    data_person$Education == "4 year degree" ~ 1,
    data_person$Education == "2 year degree" ~ 0,
    data_person$Education == "Some college" ~ 0,
    data_person$Education == "High School" ~ 0,
    data_person$Education == "Less than high school" ~ 0)
```

Gender

```r
#Creating dummy variable for Gender. Male = 1, else = 0.
data_person$male_num <- ifelse(data_person$Gender == "Male",1,0)
```

AI Trustworthiness Rating

```r
# AI TRUST - numeric version
data_person$AI_trust_num <- case_when(
  #case_when: if response is equal to "x", assign "y"
  data_person$AI_trust == "Very untrustworthy" ~ 1,
  data_person$AI_trust == "Somewhat untrustworthy" ~ 2,
  data_person$AI_trust == "Neither untrustworthy nor trustworthy" ~ 3,
  data_person$AI_trust == "Somewhat trustworthy" ~ 4,
  data_person$AI_trust == "Very trustworthy" ~ 5,
)
```

Survey Task Difficulty Rating

```r
# TASK DIFFICULTY - numeric version
data_person$Task_diff_num <- case_when(
  #case_when: if response is equal to "x", assign "y"
  data_person$Task_diff == "Extremely easy" ~ 1,
  data_person$Task_diff == "Somewhat easy" ~ 2,
  data_person$Task_diff == "Neither easy nor difficult" ~ 3,
  data_person$Task_diff == "Somewhat difficult" ~ 4,
  data_person$Task_diff == "Extremely difficult" ~ 5,
)
```

Animals Domian Knowledge Self-Rating

```r
#Domain Knowledge - Animals - Numeric
data_person$Dmn_know_a_num <- case_when(
  #case_when: if response is equal to "x", assign "y"
  data_person$Dmn_know_a == "Extremely well" ~ 1,
  data_person$Dmn_know_a == "Very well" ~ .8,
  data_person$Dmn_know_a == "Moderately well" ~ .6,
  data_person$Dmn_know_a == "Somewhat well" ~ .4,
  data_person$Dmn_know_a == "Slightly well" ~ .2,
  data_person$Dmn_know_a == "Not well at all" ~ 0
)
```

Plants Domain Knowledge Self-Rating

Domain knowledge ratings of plants are represented in binary terms since approximately half the participants rated they had no domain knowledge in identifying plants.

```
#Domain knowledge - Plants - Binary
data_person$Dmn_know_p_num <- case_when(
  #case_when: if response is equal to "x", assign "y"
  data_person$Dmn_know_p == "Extremely well" ~ 1,
  data_person$Dmn_know_p == "Very well" ~ .8,
  data_person$Dmn_know_p == "Moderately well" ~ .6,
  data_person$Dmn_know_p == "Somewhat well" ~ .4,
  data_person$Dmn_know_p == "Slightly well" ~ .2,
  data_person$Dmn_know_p == "Not well at all" ~ 0
)
```

Attention Checks

```
#Howler Monkey is the correct response to the first attention check.
data_person$atn_ch1_num <- #creating a new column for attention check 1.
  ifelse(data_person$atn_ch1 == "Howler Monkey",1, 0)

#Attention check asking how many recommendations they get for each image. "6"
data_person$atn_ch2_num <-  #creating column for attention check 2.
  ifelse(data_person$atn_ch2 == 6,1, 0) #If they chose 6, apply 1 (correct),
                                        #else 0.

#Attention check mentions to pick "Howler Monkey".
data_person$atn_ch3_num <- #creating column for attention check 3.
  ifelse(data_person$atn_ch3 == "Howler Monkey",1,0)

# sum to indicate total attention checks correctly responded to
data_person$atn_ch_sum <- rowSums(data_person %>% #summing three columns
                                        select(atn_ch1_num:atn_ch3_num))

# binary indicator for passing all attention checks.
data_person$atn_ch <- #if they passed all three checks, assign 1, else 0.
  ifelse(data_person$atn_ch1_num == 1 &
           data_person$atn_ch2_num == 1 &
           data_person$atn_ch3_num == 1,1, 0)
```

Tidying data to help reshaping the tibble

Storing column names alphabetically in a vector so that it can be rearranged in a desired manner for analysis. Chunk below is for *Stimuli Responses.*

```
#Rearranging columns - responses to no AI stimuli.
no_AI_resp <- data_person %>% #store in a vector
```

```
  select(contains("no_AI_resp")) %>%
  colnames() %>% #choose columns with NO_AI_resp in the name
  sort() #arrange alphabetically

#Rearranging columns - responses to AI_nobar stimuli.
AI_nobar_resp <- data_person %>% #store in a vector
  select(contains("AI_nobar_resp")) %>%
  colnames() %>% #choose columns with AI_nobar_resp in the name
  sort() #arrange alphabetically

#Rearranging columns - responses to AI_bar stimuli.
AI_bar_resp <- data_person %>% #store in a vector
  select(contains("AI_bar_resp")) %>%
  colnames() %>% #choose columns with AI_bar_resp in the name
  sort() #arrange alphabetically
```

Storing column names into vectors for *confidence ratings*.

```
#Rearranging columns - responses to no AI Confidence.
no_AI_conf <- data_person %>%
  select(contains("no_AI_conf")) %>%
  colnames() %>%
  sort()

#Rearranging columns - responses to AI_nobar Confidence.
AI_nobar_conf <- data_person %>%
  select(contains("AI_nobar_conf")) %>%
  colnames() %>%
  sort()

#Rearranging columns - responses to AI_bar Confidence.
AI_bar_conf <- data_person %>%
  select(contains("AI_bar_conf")) %>%
  colnames() %>%
  sort()
```

Storing column names into vectors for *time taken per image*.

```
#Rearranging columns - time taken in no AI condition.
no_AI_time <- data_person %>%
  select(contains("no_AI_time")) %>%
  colnames() %>%
  sort()

#Rearranging columns - time taken in AI_nobar condition.
AI_nobar_time <- data_person %>%
  select(contains("AI_nobar_time")) %>%
  colnames() %>%
  sort()

#Rearranging columns - time taken in AI_bar condition.
AI_bar_time <- data_person %>%
  select(contains("AI_bar_time")) %>%
  colnames() %>%
  sort()
```

Storing column names into vectors for *AI Usefulness ratings*.

```
#Rearranging columns - Usefulness rating in AI_nobar condition.
AI_nobar_use <- data_person %>%
  select(contains("AI_nobar_use")) %>%
  colnames() %>%
  sort()

#Rearranging columns - Usefulness rating in AI_bar condition.
AI_bar_use <- data_person %>%
  select(contains("AI_bar_use")) %>%
  colnames() %>%
  sort()
```

Reshaping Data frame

```
data_person <- data_person %>%
  select(no_AI_resp, AI_nobar_resp, AI_bar_resp, no_AI_conf, AI_nobar_conf,
         AI_bar_conf, no_AI_time, AI_nobar_time, AI_bar_time, AI_nobar_use,
         AI_bar_use, #arranging the dataframe by using the vectors
         Group, everything()) #mentioning everything else to remain untouched.
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(no_AI_resp)` instead of `no_AI_resp` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_resp)` instead of `AI_nobar_resp` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_resp)` instead of `AI_bar_resp` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(no_AI_conf)` instead of `no_AI_conf` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_conf)` instead of `AI_nobar_conf` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_conf)` instead of `AI_bar_conf` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(no_AI_time)` instead of `no_AI_time` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_time)` instead of `AI_nobar_time` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_time)` instead of `AI_bar_time` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_use)` instead of `AI_nobar_use` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_use)` instead of `AI_bar_use` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

Time taken per stimuli

```r
#converting to numeric
time_chr_to_num <- function(x){
  as.numeric(x)
}

#Applying function to all stimulus' time recordings
data_person <- data_person %>% #choosing tibble
  mutate_at(.vars = vars(`Barc_no_AI_time_Page Submit` :
                          `Zuch_AI_bar_time_Page Submit`),
            #.vars - mentioning the column names to mutate
            .funs = funs(time_chr_to_num)) #applying the function
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

Stimuli Correct Responses

```r
source("stimuli_response_coding.R") #code is included in a separate R file.
```

Confidence ratings for each stimuli

```r
#creating a function to convert chr to numerics.
conf_num <- function(x){
case_when(
  #case_when: if response is equal to "x", assign "y"
  x == "Not confident at all (0% - 20%)" ~ 0,
  x == "Slightly confident (21% - 40%)" ~ .2,
  x == "Moderately confident (41% - 60%)" ~ .4,
  x == "Very confident    (61% - 80%)" ~ .6,
  x == "Extremely confident (81% - 99%)" ~ .8,
  x ==  "Absolutely confident (100%)" ~ 1)}

#applying the function in tibble.
data_person <- data_person %>% #choosing tibble
  mutate_at(.vars = vars(Barc_no_AI_conf: Zuch_AI_bar_conf),
            #.vars - mentioning the column names to mutate
            .funs = funs(conf_num)) #applying the function

#converting to numeric
conf_chr_to_num <- function(x){
  as.numeric(x)
}

#applying function to all confidence responses
data_person <- data_person %>% #choosing tibble
  mutate_at(.vars = vars(Barc_no_AI_conf : Zuch_AI_bar_conf),
            #.vars - mentioning the column names to mutate
            .funs = funs(conf_chr_to_num)) #applying the function
```

AI usefulness ratings

```
#Creating a function for AI usefulness columns.
AI_use_num <- function(x){
  case_when(
  #case_when: if response is equal to "x", assign "y"
  x == "Extremely useful" ~ 1,
  x == "Very useful" ~ .8,
  x == "Moderately useful" ~ .6,
  x == "Somewhat useful" ~ .4,
  x == "Slightly useful" ~ .2,
  x == "Not useful at all" ~ 0)}

#applying function - converting them to numbers
data_person <- data_person %>% #choosing tibble
  mutate_at(.vars = vars(Barc_AI_nobar_use:Zuch_AI_bar_use),
            #.vars - mentioning the column names to mutate
            .funs = funs(AI_use_num,)) #applying the function
```

Total time taken

```
#Converting to numeric
data_person$`Duration (in seconds)` <- as.numeric(
                                       data_person$`Duration (in seconds)`
                                       )
```

END OF GENERAL CLEANING OF VARIABLES

ALL FOLLOWING CODE CHUNKS BELOW ARE FOR CREATING THE MULITPLE DATASETS

Separating Plants and Animals stimulus.

Creating two data frames that will separate the plants and animals stimulus so that data can be analyzed how the performance in two different domains.

```
plants_person <- data_person %>%
  select(contains(c("Cherry", "CPoppy", "Cyclmn", "Eldb", "FlyOrc", "Frag",
                    "Gazn", "Hen", "Huis", "Log", "Mulb", "NZB", "Pars",
                    "Phaius", "Polyp", "Pomg", "Rseed", "Shal", "Zuch")),
         c("Group":"atn_ch"))


animals_person <- data_person %>%
  select(-contains(c("Cherry", "CPoppy", "Cyclmn", "Eldb", "FlyOrc", "Frag",
                    "Gazn", "Hen", "Huis", "Log", "Mulb", "NZB", "Pars",
                    "Phaius", "Polyp", "Pomg", "Rseed", "Shal", "Zuch")),
         c("Group":"atn_ch"))
```

TIDYING FOR PLANTS DATASET

Tidying data to help reshaping the tibble

Storing column names alphabetically in a vector so that it can be rearranged in a desired manner for analysis. Chunk below is for *Stimuli Responses*.

```r
#Rearranging columns - responses to no AI stimuli.
no_AI_resp_p <- plants_person %>% #store in a vector
  select(contains("no_AI_resp")) %>%
  colnames() %>% #choose columns with NO_AI_resp_p in the name
  sort() #arrange alphabetically

#Rearranging columns - responses to AI_nobar stimuli.
AI_nobar_resp_p <- plants_person %>% #store in a vector
  select(contains("AI_nobar_resp")) %>%
  colnames() %>% #choose columns with AI_nobar_resp_p in the name
  sort() #arrange alphabetically

#Rearranging columns - responses to AI_bar stimuli.
AI_bar_resp_p <- plants_person %>% #store in a vector
  select(contains("AI_bar_resp")) %>%
  colnames() %>% #choose columns with AI_bar_resp_p in the name
  sort() #arrange alphabetically
```

Storing column names into vectors for *confidence ratings*.

```r
#Rearranging columns - responses to no AI Confidence.
no_AI_conf_p <- plants_person %>%
  select(contains("no_AI_conf")) %>%
  colnames() %>%
  sort()

#Rearranging columns - responses to AI_nobar Confidence.
AI_nobar_conf_p <- plants_person %>%
  select(contains("AI_nobar_conf")) %>%
  colnames() %>%
  sort()

#Rearranging columns - responses to AI_bar Confidence.
AI_bar_conf_p <- plants_person %>%
  select(contains("AI_bar_conf")) %>%
  colnames() %>%
  sort()
```

Storing column names into vectors for *time taken per image*.

```r
#Rearranging columns - time taken in no AI condition.
no_AI_time_p <- plants_person %>%
  select(contains("no_AI_time")) %>%
  colnames() %>%
  sort()

#Rearranging columns - time taken in AI_nobar condition.
AI_nobar_time_p <- plants_person %>%
  select(contains("AI_nobar_time")) %>%
  colnames() %>%
  sort()

#Rearranging columns - time taken in AI_bar condition.
```

```r
AI_bar_time_p <- plants_person %>%
  select(contains("AI_bar_time")) %>%
  colnames() %>%
  sort()
```

Storing column names into vectors for *AI Usefulness ratings*.

```r
#Rearranging columns - Usefulness rating in AI_nobar condition.
AI_nobar_use_p <- plants_person %>%
  select(contains("AI_nobar_use")) %>%
  colnames() %>%
  sort()

#Rearranging columns - Usefulness rating in AI_bar condition.
AI_bar_use_p <- plants_person %>%
  select(contains("AI_bar_use")) %>%
  colnames() %>%
  sort()
```

Reshaping Data frame

```r
plants_person <- plants_person %>%
  select(no_AI_resp_p, AI_nobar_resp_p, AI_bar_resp_p, no_AI_conf_p, AI_nobar_conf_p,
         AI_bar_conf_p, no_AI_time_p, AI_nobar_time_p, AI_bar_time_p, AI_nobar_use_p,
         AI_bar_use_p, #arranging the dataframe by using the vectors
         Group, everything()) #mentioning everything else to remain untouched.
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(no_AI_resp_p)` instead of `no_AI_resp_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_resp_p)` instead of `AI_nobar_resp_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_resp_p)` instead of `AI_bar_resp_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(no_AI_conf_p)` instead of `no_AI_conf_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_conf_p)` instead of `AI_nobar_conf_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_conf_p)` instead of `AI_bar_conf_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(no_AI_time_p)` instead of `no_AI_time_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_time_p)` instead of `AI_nobar_time_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_time_p)` instead of `AI_bar_time_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_nobar_use_p)` instead of `AI_nobar_use_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.

## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(AI_bar_use_p)` instead of `AI_bar_use_p` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

CLEANING FOR PLANTS BY PERSON DATASET

EACH OBSERVATION IS ONE PERSON

Averaging across images for plants data set

Creating/Finalizing the data set where each observation is a person. Chunk below creates all the *AI vs No-AI averages* across images.

```r
#Average Accuracy no-AI
plants_person$avg_no_AI_acc <- (rowSums(plants_person %>%
            #summing columns for all images in NO AI condition
            select(Cherry_no_AI_resp : Zuch_no_AI_resp), na.rm = TRUE))/19
            #Divided by 19 since there are 19 stimulus in the experiment

#Average Accuracy AI
plants_person$avg_AI_acc <- (rowSums(plants_person %>%
            select(Cherry_AI_nobar_resp : Zuch_AI_bar_resp), na.rm = TRUE))/19

#Average Confidence no-AI
plants_person$avg_no_AI_conf <- (rowSums(plants_person %>%
            select(Cherry_no_AI_conf : Zuch_no_AI_conf), na.rm = TRUE))/19

#Average Confidence AI
plants_person$avg_AI_conf <- (rowSums(plants_person %>%
            select(Cherry_AI_nobar_conf : Zuch_AI_bar_conf), na.rm = TRUE))/19

#Average Time per image no-AI
plants_person$avg_no_AI_time <- (rowSums(plants_person %>%
            select(`Cherry_no_AI_time_Page Submit` :
                     `Zuch_no_AI_time_Page Submit`), na.rm = TRUE))/19

#Average Time per image AI
plants_person$avg_AI_time <- (rowSums(plants_person %>%
            select(`Cherry_AI_nobar_time_Page Submit` :
                     `Zuch_AI_bar_time_Page Submit`), na.rm = TRUE))/19

#Placeholder for AI usefulness rating in No-AI condition.
plants_person$avg_no_AI_use <- NA

#Average AI usefulness rating.
plants_person$avg_AI_use <- (rowSums(plants_person %>%
            select(Cherry_AI_nobar_use : Zuch_AI_bar_use), na.rm = TRUE))/19
```

Data set containing only AI vs NO-AI columns

```
#Each observation is a person. No mixed effects to be modeled with this dataset
plants_by_person <- select(plants_person, c("Group","StartDate":"ResponseId", "PROLIFIC_PID" : "avg_AI_
```

Stacking columns - all accuracy in one column, all conf in one column. . .

```
plants_by_person <- cbind(plants_by_person %>% #choosing dataset
                     select(1:18), #mentioning columns to be left untouched.
                   stack(plants_by_person %>% #makes dataset from wide to long
                           select(avg_no_AI_acc:avg_AI_acc)), #choosing cols.
                   stack(plants_by_person %>%
                           select(avg_no_AI_conf:avg_AI_conf)),
                   stack(plants_by_person %>%
                           select(avg_no_AI_time:avg_AI_time)),
                   stack(plants_by_person %>%
                           select(avg_no_AI_use:avg_AI_use)))
```

```
#Providing names for the newly stacked columns
colnames(plants_by_person)[19:26] <- c("accuracy", "exp_condition",
                                "confidence", "conf_name",
                                "time_taken", "time_name",
                                "AI_use", "AI_use_name")

#Since they are repeated, columns are removed
plants_by_person <- plants_by_person %>%
  select(-contains("name"))
```

```
#Creating a column that indicates if the response was for No-AI or AI condition
plants_by_person$AI <- ifelse(plants_by_person$exp_condition == "avg_AI_acc", 1, 0)

#Column to indicate if uncertainty information was provided
plants_by_person$bar <- ifelse(plants_by_person$AI == 1 & #both statements must be true
                          plants_by_person$Group == "AI_bars" ,1, 0)
#if AI information was provied & they were placed in AI_bars condition, then 1.
```

```
#removing columns that are not necessary anymore
plants_by_person <- select(plants_by_person, -c("Group":"Finished","PROLIFIC_PID", "atn_ch1_num":"atn_c
```

Exporting cleaned *by_person averages* data set

```
#2 Each observation is one person for plants
write.csv(plants_by_person,file="Datasets/Data_cleaned_plants_person.csv")
#Use only for t-tests and linear regressions with no mixed effects.
#This dataset will be used for the first part of the analysis.
```

END OF CLEANING FOR PLANTS BY PERSON DATASET - ONLY PLANTS

EACH OBSERVATION IS ONE PERSON

CLEANING FOR ANIMALS BY PERSON DATASET

EACH OBSERVATION IS ONE PERSON

Averaging across images for animals data set

Creating/Finalizing the data set where each observation is a person. Chunk below creates all the *AI vs No-AI averages* across images.

```
#Average Accuracy no-AI
animals_person$avg_no_AI_acc <- (rowSums(animals_person %>%
            #summing columns for all images in NO AI condition
            select(Barc_no_AI_resp : Smonk_no_AI_resp), na.rm = TRUE))/13
            #Divided by 13 since there are 13 stimulus in the experiment

#Average Accuracy AI
animals_person$avg_AI_acc <- (rowSums(animals_person %>%
            select(Barc_AI_nobar_resp : Smonk_AI_bar_resp), na.rm = TRUE))/13

#Average Confidence no-AI
animals_person$avg_no_AI_conf <- (rowSums(animals_person %>%
            select(Barc_no_AI_conf : Smonk_no_AI_conf), na.rm = TRUE))/13

#Average Confidence AI
animals_person$avg_AI_conf <- (rowSums(animals_person %>%
            select(Barc_AI_nobar_conf : Smonk_AI_bar_conf), na.rm = TRUE))/13
```

```
#Average Time per image no-AI
animals_person$avg_no_AI_time <- (rowSums(animals_person %>%
            select(`Barc_no_AI_time_Page Submit` :
                     `Smonk_no_AI_time_Page Submit`), na.rm = TRUE))/13

#Average Time per image AI
animals_person$avg_AI_time <- (rowSums(animals_person %>%
            select(`Barc_AI_nobar_time_Page Submit` :
                     `Smonk_AI_bar_time_Page Submit`), na.rm = TRUE))/13

#Placeholder for AI usefulness rating in No-AI condition.
animals_person$avg_no_AI_use <- NA

#Average AI usefulness rating.
animals_person$avg_AI_use <- (rowSums(animals_person %>%
            select(Barc_AI_nobar_use : Smonk_AI_bar_use), na.rm = TRUE))/13
```

Data set containing only AI vs NO-AI columns

```
#Each observation is a person. No mixed effects to be modeled with this dataset
animals_by_person <- select(animals_person, c("Group","StartDate":"ResponseId", "PROLIFIC_PID" : "avg_AI
```

Stacking columns - all accuracy in one column, all conf in one column…

```
animals_by_person <- cbind(animals_by_person %>% #choosing dataset
                     select(1:18), #mentioning columns to be left untouched.
                   stack(animals_by_person %>% #makes dataset from wide to long
                           select(avg_no_AI_acc:avg_AI_acc)), #choosing cols.
                   stack(animals_by_person %>%
                           select(avg_no_AI_conf:avg_AI_conf)),
                   stack(animals_by_person %>%
                           select(avg_no_AI_time:avg_AI_time)),
                   stack(animals_by_person %>%
                           select(avg_no_AI_use:avg_AI_use)))
```

```
#Providing names for the newly stacked columns
colnames(animals_by_person)[19:26] <- c("accuracy", "exp_condition",
                                "confidence", "conf_name",
                                "time_taken", "time_name",
                                "AI_use", "AI_use_name")

#Since they are repeated, columns are removed
animals_by_person <- animals_by_person %>%
  select(-contains("name"))
```

```
#Creating a column that indicates if the response was for No-AI or AI condition
animals_by_person$AI <- ifelse(animals_by_person$exp_condition == "avg_AI_acc", 1, 0)

#Column to indicate if uncertainty information was provided
animals_by_person$bar <- ifelse(animals_by_person$AI == 1 & #both statements must be true
                          animals_by_person$Group == "AI_bars" ,1, 0)
#if AI information was provied & they were placed in AI_bars condition, then 1.
```

```
#removing columns that are not necessary anymore
animals_by_person <- select(animals_by_person, -c("Group":"Finished","PROLIFIC_PID", "atn_ch1_num":"atn
```

Exporting cleaned *by_person averages* data set

```r
#4 Each observation is one person for animals stimuli
write.csv(animals_by_person,file="Datasets/Data_cleaned_animals_person.csv")
#Use only for t-tests and linear regressions with no mixed effects.
#This dataset will be used for the first part of the analysis.
```

END OF CLEANING FOR ANIMALS BY PERSON DATASET - ONLY ANIMALS

EACH OBSERVATION IS ONE PERSON

Averaging across images - BOTH PLANTS AND ANIMALS

Creating/Finalizing the data set where each observation is a person. Chunk below creates all the *AI vs No-AI averages* across images.

```r
#Average Accuracy no-AI
data_person$avg_no_AI_acc <- (rowSums(data_person %>%
            #summing columns for all images in NO AI condition
            select(Barc_no_AI_resp : Zuch_no_AI_resp), na.rm = TRUE))/32
            #Divided by 32 since there are 32 stimulus in the experiment

#Average Accuracy AI
data_person$avg_AI_acc <- (rowSums(data_person %>%
            select(Barc_AI_nobar_resp : Zuch_AI_bar_resp), na.rm = TRUE))/32

#Average Confidence no-AI
data_person$avg_no_AI_conf <- (rowSums(data_person %>%
            select(Barc_no_AI_conf : Zuch_no_AI_conf), na.rm = TRUE))/32

#Average Confidence AI
data_person$avg_AI_conf <- (rowSums(data_person %>%
            select(Barc_AI_nobar_conf : Zuch_AI_bar_conf), na.rm = TRUE))/32

#Average Time per image no-AI
data_person$avg_no_AI_time <- (rowSums(data_person %>%
            select(`Barc_no_AI_time_Page Submit` :
                   `Zuch_no_AI_time_Page Submit`), na.rm = TRUE))/32

#Average Time per image AI
data_person$avg_AI_time <- (rowSums(data_person %>%
            select(`Barc_AI_nobar_time_Page Submit` :
```

```
                      `Zuch_AI_bar_time_Page Submit`), na.rm = TRUE))/32

#Placeholder for AI usefulness rating in No-AI condition.
data_person$avg_no_AI_use <- NA

#Average AI usefulness rating.
data_person$avg_AI_use <- (rowSums(data_person %>%
            select(Barc_AI_nobar_use : Zuch_AI_bar_use), na.rm = TRUE))/32
```

Data set containing only AI vs NO-AI columns

```
#Each observation is a person. No mixed effects to be modeled with this dataset
by_person <- select(data_person, c("Group","StartDate":"ResponseId", "PROLIFIC_PID" : "avg_AI_use"))
```

Stacking columns - all accuracy in one column, all conf in one column…

```
by_person <- cbind(by_person %>% #choosing dataset
                      select(1:18), #mentioning columns to be left untouched.
                    stack(by_person %>% #makes dataset from wide to long
                            select(avg_no_AI_acc:avg_AI_acc)), #choosing cols.
                    stack(by_person %>%
                            select(avg_no_AI_conf:avg_AI_conf)),
                    stack(by_person %>%
                            select(avg_no_AI_time:avg_AI_time)),
                    stack(by_person %>%
                            select(avg_no_AI_use:avg_AI_use)))
```

```
#Providing names for the newly stacked columns
colnames(by_person)[19:26] <- c("accuracy", "exp_condition",
                                "confidence", "conf_name",
                                "time_taken", "time_name",
                                "AI_use", "AI_use_name")

#Since they are repeated, columns are removed
by_person <- by_person %>%
  select(-contains("name"))
```

```
#Creating a column that indicates if the response was for No-AI or AI condition
by_person$AI <- ifelse(by_person$exp_condition == "avg_AI_acc", 1, 0)

#Column to indicate if uncertainty information was provided
by_person$bar <- ifelse(by_person$AI == 1 & #both statements must be true
                          by_person$Group == "AI_bars" ,1, 0)
#if AI information was provied & they were placed in AI_bars condition, then 1.
```

```
#removing columns that are not necessary anymore
by_person <- select(by_person, -c("Group":"Finished","PROLIFIC_PID", "atn_ch1_num":"atn_ch_sum", "exp_c
```

Exporting cleaned *by_person averages* data set

```
#6 Each observation is one person for both plants and animals
write.csv(by_person,file="Datasets/Data_cleaned_person.csv")
#Use only for t-tests and linear regressions with no mixed effects.
#This dataset will be used for the first part of the analysis.
```

END OF CLEANING FOR PERSON DATASET - BOTH PLANTS AND ANIMALS

EACH OBSERVATION IS ONE PERSON