Supplemental file

Last updated 2021-08-24

Contents

1 Cleaning

The current section documents the data cleaning process.

1.1 Workspace

```
library(here) # for working with files
library(tidyverse) # for cleaning
library(janitor) # for variable names
library(stringi) # for generating random strings
library(lme4) # for multilevel modeling
library(lmerTest) # for p-values
library(sjPlot) # for figures
library(ggpubr) # for prettier plots
library(kableExtra) # for nicer tables
library(stringdist) # for scoring memory task
library(papaja) # for pretty numbers
library(psych) # for correlation tests
library(broom.mixed) # for tidying mulitilevel models
```

1.2 Change participant ID values

Before we begin, we create new versions of each data_t1 file that can be shared for purposes of reproducibility. These data_t1 files do not include variables that contain potentially identifying meta-data_t1 (e.g., IP address, latitude and longitude). Importantly, we also replace all Prolific ID values with new, random strings, to prevent the possibility that these participants are later identified. We also fix an error that can be introduced through Qualtrics, specifically that all or parts of the text string "Value will be set from panel or URL" is sometimes entered into the text box for ID. Prolific ID values are always 24 characters long and start with a number – we search for strings that meet this criteria.

(We note that the code chunks in this subsection are turned off in the RMarkdown file – `eval = F` – as readers will not be able to run these chunks.)

```
# function to load raw file, clean the names, and remove meta-data_t1
# creating a function ensures the same procedure is applied to all
# orginal datasets

load_data = function(path){

  full_path = here(path)
  data_labels = read_csv(full_path)
  data_obj = read_csv(full_path,
                  skip = 3,
                  col_names = names(data_labels))

  data_obj = clean_names(data_obj)

  data_obj = data_obj %>%
    select(-end_date,
           -ip_address,
           -progress,
```

```
            -finished,
            -recorded_date,
            -status,
            -response_id,
            -external_reference,
            -distribution_channel,
            -user_language,
            -starts_with("recipient"),
            -starts_with("location"),
            -starts_with("meta_info"),
            -prolific_pid)

  #this fixes a column naming error in dataset 2B
  if(!("proid" %in% names(data_obj))) names(data_obj) = str_replace(names(data_obj), "q763", "proid")

  data_obj = data_obj %>%
    mutate(proid = str_extract(proid, "\\d([[:alnum:]]{23})"))

  return(data_obj)
}

data_t1 <- load_data("data/Wording_July 13, 2021_20.00.text.csv")
data_2A <- load_data("data/Wording 2A_August 13, 2021_14.49.text.csv")
data_2B <- load_data("data/Wording 2B_August 4, 2021_18.49.text.csv")
data_2C <- load_data("data/Wording 2C_August 3, 2021_18.02.csv")
data_2D <- load_data("data/Wording 2D_July 29, 2021_14.55.text.csv")
```

Next, we identify all unique participant IDs. For each, we generate a new string, Then we replace the original ID values with the new strings.

```
original_id <- unique(c(data_t1$proid,
                        data_2A$proid,
                        data_2B$proid,
                        data_2C$proid,
                        data_2D$proid))

#remove missing values -- represent bots or tests
original_id = original_id[!is.na(original_id)]

#generate new ids (randoms tring of letters and numbers)
set.seed(202108)
new_id <- stri_rand_strings(n = length(original_id), length = 24)

#replace old string with new string
for(i in 1:length(original_id)){
  data_t1$proid[data_t1$proid == original_id[i]] <- new_id[i]
  data_2A$proid[data_2A$proid == original_id[i]] <- new_id[i]
  data_2B$proid[data_2B$proid == original_id[i]] <- new_id[i]
  data_2C$proid[data_2C$proid == original_id[i]] <- new_id[i]
  data_2D$proid[data_2D$proid == original_id[i]] <- new_id[i]
}
```

We end by saving each data_t1 frame as new .csv files, to be uploaded to OSF and shared for reproduction.

```
write_csv(data_t1, file = here("deidentified data/data_time1.csv"))
write_csv(data_2A, file = here("deidentified data/data_time2_A.csv"))
write_csv(data_2B, file = here("deidentified data/data_time2_B.csv"))
write_csv(data_2C, file = here("deidentified data/data_time2_C.csv"))
write_csv(data_2D, file = here("deidentified data/data_time2_D.csv"))
```

1.3 Time 1

We load the deidentified Time 1 data here.

```
data_t1 <- read_csv(here("deidentified data/data_time1.csv"))
```

We rename several columns, in order to facilitate the use of regular expressions later. Specifically, we remove the underscores (_) in the columns pertaining to broad-mindedness and self-disciplined.

```
names(data_t1) = str_replace(names(data_t1), "broad_mind", "broadmind")
names(data_t1) = str_replace(names(data_t1), "self_disciplind", "selfdisciplined")
```

We can also remove the meta-data (timing, etc) around two attention check adjectives, "human" and "asleep".

```
data_t1 = data_t1 %>%
  select(-starts_with("t_human"),
         -starts_with("t_asleep"))
```

1.3.1 Recode personality item responses to numeric

We recode the responses to personality items, which we downloaded as text strings. We chose to use text strings as opposed to numbers to avoid any possibility that the Qualtrics-set coding was incorrect. We start this process by identifying the personality items (`p_items`) using regular expressions. All personality items take a format like `outgoing_a` or `helpful_b_2`; that is, they start with the adjective, followed by a letter indicating with which condition or item format the adjective was presented, and sometimes they are followed by a 2, indicating it was the second time the participant saw the adjective. We can represent this pattern using regular expressions.

```
p_items = str_extract(names(data_t1), "^[[:alpha:]]*_[abcd](_2)?$")
p_items = p_items[!is.na(p_items)]

personality_items = select(data_t1, proid, all_of(p_items))
```

Next, we write a simple function to recode values. We find the `case_when` function to be the most clear method of communicating the recoding process when moving from string to numeric.

```
recode_p = function(x){
  y = case_when(
    x == "Very inaccurate" ~ 1,
    x == "Moderately inaccurate" ~ 2,
    x == "Slightly inaccurate" ~ 3,
    x == "Slightly accurate" ~ 4,
    x == "Moderately accurate" ~ 5,
    x == "Very accurate" ~ 6,
```

```
    TRUE ~ NA_real_)
  return(y)
}
```

Finally, we apply this function to all personality items.

```
personality_items = personality_items %>%
  # apply to all varibles except proid
  mutate(across(!c(proid), recode_p))
```

Now we merge the recoded values back into the data_t1.

```
# remove personality items from data file
data_t1 = select(data_t1, -all_of(p_items))
# merge in recoded personality items
data_t1 = full_join(data_t1, personality_items)
```

1.3.2 Drop bots and inattentive participants

1.3.2.1 Based on ID Recall that when preparing the data files for sharing, we replaced all Prolific IDs with random strings. A consequence of this cleaning is that any ID entered that did not have a string meeting the Prolific ID format requirements (24 character, starting with a number) was replaced with `NA`. To remove these bots, we can simply filter out missing ID values.

We removed 9 participants without valid Prolific IDs.

```
data_t1 = data_t1 %>%
  filter(english %in% c("Well", "Very well (fluent/native)"))
```

1.3.2.2 Based on language We removed 0 participants that do not speak english well or very well.

1.3.2.3 Based on patterns We remove any participant who provides the same response to over half of the items (17 or more items) from a given block in a row.

To proceed, first we create a dataframe containing just the responses to personality items in the first block.

```
# first, identify unique adjectives, in order
adjectives = p_items %>%
  str_remove_all("_.") %>%
  unique()

# extract block 1 questions using regular expressions
# these follow the personality item format described above, but never end with 2
block1 = data_t1 %>%
  select(proid, matches("^[[:alpha:]]+_[abcd]$"))
```

Next, we rename the variables. Instead of variable names identifying the specific adjective (e.g., outgoing_a), we need variable names which indicate the order in which the adjective was seen by the participant (e.g., trait01_a). This will help us determine patterns by item order, rather than adjective content. Participants all saw adjectives in the same order (i.e., all participants, regardless of condition, saw outgoing first).

```
#rename variables
n = 0
for(i in adjectives){ # for each adjective
  n = n+1 # identify its location in the presentation
  names(block1) = str_replace(names(block1), #in variable names
                              # replace the adjective string
                              i,
                              # with the word trait followed by its place
                              paste0("trait", str_pad(n, 2, pad = "0")))
}
```

We use `gather` and `spread` to quickly combine columns measuring the same trait. That is, instead of having columns trait01_a, trait01_b, trait01_c, and trait01_d, we now have a single column called trait01.

```
block1 = block1 %>%
  gather(item, response, -proid) %>%
  filter(!is.na(response)) %>%
  separate(item, into = c("item", "format")) %>%
  select(-format) %>%
  spread(item, response)
```

To count the number of runs, we loop through participants and, within participant, loop through columns. Within participant, we create an object called `run`. If a response to a personality item is the same as the participant's response to the previous item, we increase the value of `run` by 1. If this new value is the largest `run` value for that participant, it becomes the value of an object called `maxrun`. If the participant gives a new response, `run` is reset to 0. We record the `maxrun` value for each partipant in a variable called `block1_runs`.

```
block1_runs = numeric(length = nrow(block1))


for(i in 1:nrow(block1)){
  run = 0
  maxrun = 0
  for(j in 3:ncol(block1)){
    if(block1[i,j] == block1[i, j-1]){
      run = run+1
      if(run > maxrun) maxrun = run
      } else{ run = 0}
  }
  block1_runs[i] = maxrun
}

#add to data_t1 frame
block1$block1_runs = block1_runs
```

Here we repeat the process described above with Block 2 data.

```
# extract block 2 questions
block2 = data_t1 %>%
  select(proid, matches("^[[:alpha:]]+_[abcd]_2$"))

#rename variables
```

```
n = 0
for(i in adjectives){
  n = n+1
  names(block2) = str_replace(names(block2), i, paste0("trait", str_pad(n, 2, pad = "0")))
}

block2 = block2 %>%
  gather(item, response, -proid) %>%
  filter(!is.na(response)) %>%
  mutate(item = str_remove(item, "_2")) %>%
  separate(item, into = c("item", "format")) %>%
  select(-format) %>%
  spread(item, response)

block2_runs = numeric(length = nrow(block2))

#identify max run for each participant
for(i in 1:nrow(block2)){
  run = 0
  maxrun = 0
  for(j in 3:ncol(block2)){
    if(block2[i,j] == block2[i, j-1]){
      run = run+1
      if(run > maxrun) maxrun = run
      } else{ run = 0}
  }
  block2_runs[i] = maxrun
}

#add to data_t1 frame
block2$block2_runs = block2_runs
```

We combine the variables holding the maximum runs into a single data frame. We will remove participants if their maximum run in either block was greater than or equal to 17. See Figure 1 for a visualization of the spread and associations between run lengths across participants.

```
#combine results
runs_data = block1 %>%
  select(proid, block1_runs) %>%
  full_join(select(block2, proid, block2_runs)) %>%
  mutate(
    remove = case_when(
      block1_runs >= 17 ~ "Remove",
      block2_runs >= 17 ~ "Remove",
      TRUE ~ "Keep"
    ))
```

There were 2 participants who provided the same answer 17 or more times in a row. These participants were removed from the analyses.

```
data_t1 = data_t1 %>%
  full_join(select(runs_data, proid, remove)) %>%
  filter(remove != "Remove") %>%
```

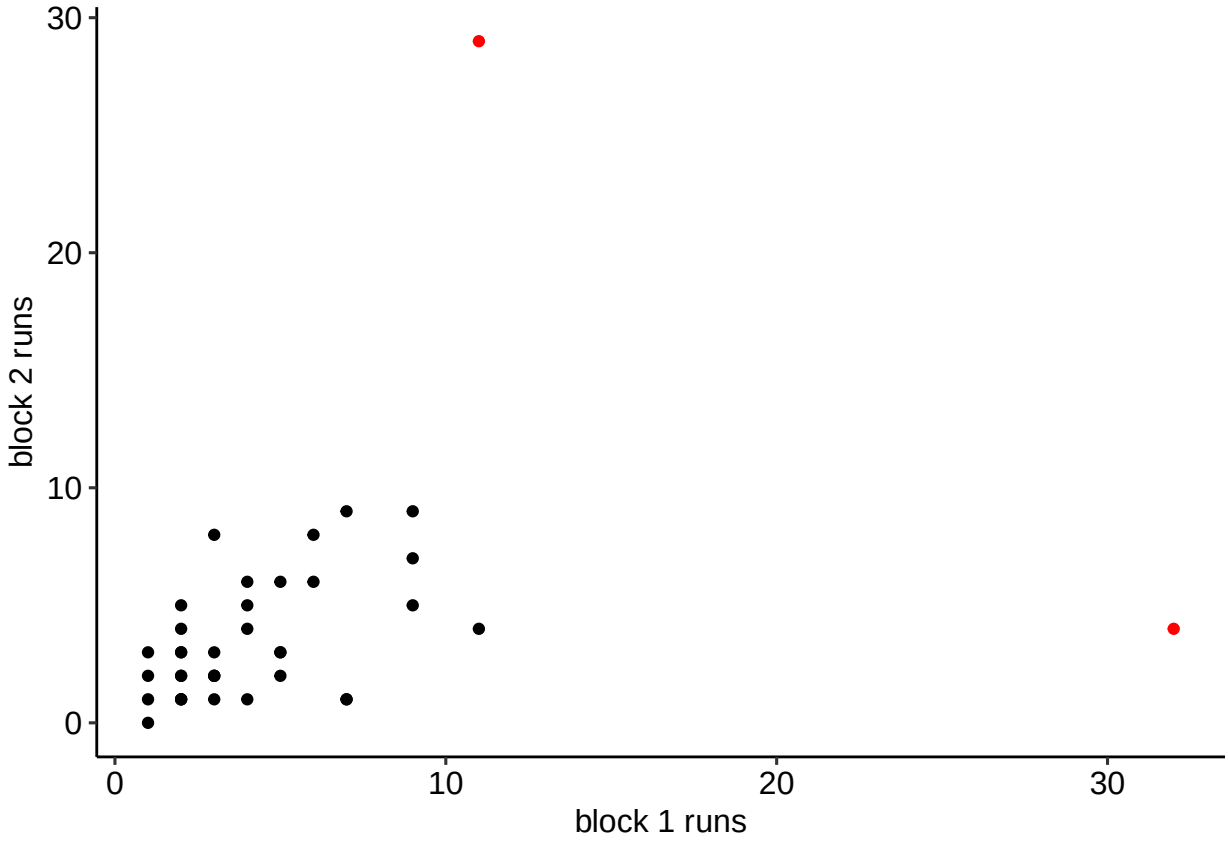


Figure 1: Maximum number of same consecutive responses in personality blocks.

```
  select(-remove)

rm(runs_data)
```

1.3.2.4 Based on inattentive responding

We expect to exclude any participant who has an average response of 4 ("slightly agree") or greater to the attention check items. Two items from the Inattentive and Deviant Responding Inventory for Adjectives (IDRIA) scale (Kay & Saucier, in prep) have been included here, in part to help evaluate the extent of inattentive responding but also to consider the effect of item wording on these items. The two items used here (i.e., "Asleep", "Human") were chosen to be as inconspicuous as possible, so as to not to inflate item response duration. The frequency item (i.e., "human") will be reverse-scored, so that higher scores on both the infrequency and frequency items reflect greater inattentive responding. Figure 2 shows the distribution of average responses to attention check items.

```
in_average = data_t1 %>%
  # reverse score human
  mutate(across(matches("^human"),  ~(.x*-1)+7)) %>%
  # select id and attention check items
  select(proid, matches("^human"), matches("^asleep")) %>%
  gather(item, response, -proid) %>%
  filter(!is.na(response)) %>%
  group_by(proid) %>%
  summarise(avg = mean(response)) %>%
  mutate(
    remove = case_when(
      avg >= 4 ~ "Remove",
      TRUE ~ "Keep"))
```

We remove 1 participants whose responses suggest inattention.

```
data_t1 = data_t1 %>%
  full_join(select(in_average, proid, remove)) %>%
  filter(remove != "Remove") %>%
  select(-remove)
```

1.3.2.5 Based on average time to respond to personality items

First, select just the timing of the personality items. We do this by searching for specific strings: "t_[someword]*[a or b or c or d]*(maybe 2_)_page_submit."

```
timing_data = data_t1 %>%
  select(proid, matches("t_[[:alpha:]]*_[abcd](_2)?_page_submit"))
```

Next we gather into long form and remove missing timing values

```
timing_data = timing_data %>%
  gather(variable, timing, -proid) %>%
  filter(!is.na(timing))
```

To check, each participant should have the same number of responses: 62.

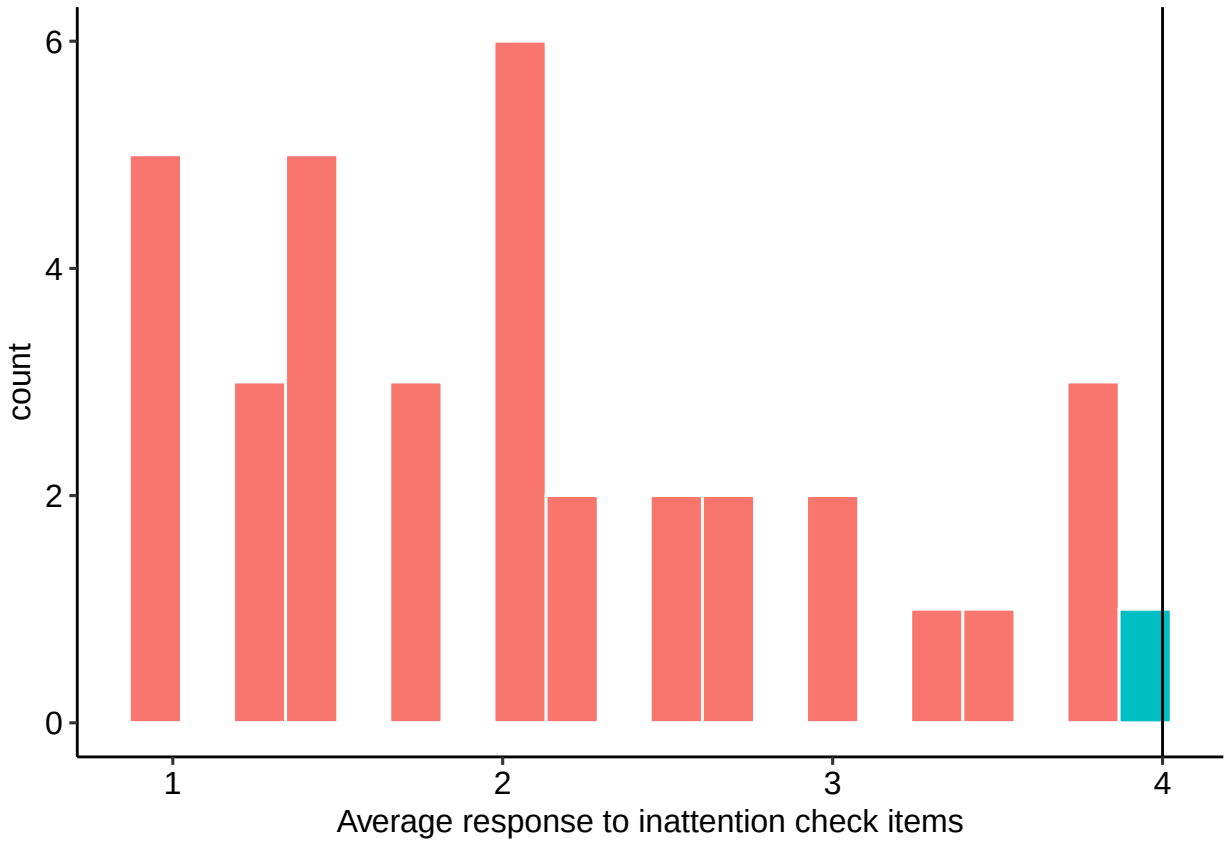


Figure 2: Average response to inattention check items

```
timing_data %>%
  group_by(proid) %>%
  count() %>%
  ungroup() %>%
  summarise(min(n), max(n))
```

```
## # A tibble: 1 x 2
##   `min(n)` `max(n)`
##      <int>    <int>
## 1       62       62
```

Excellent! Now we calculate the average response time per item for each participant. We mark a participant for removal if their average time is less than 1 second or greater than 30. See Figure 3 for a distribution of average response time.

```
timing_data = timing_data %>%
  group_by(proid) %>%
  summarise(m_time = mean(timing)) %>%
  mutate(remove = case_when(
    m_time < 1 ~ "Remove",
    m_time > 30 ~ "Remove",
    TRUE ~ "Keep"
  ))
```

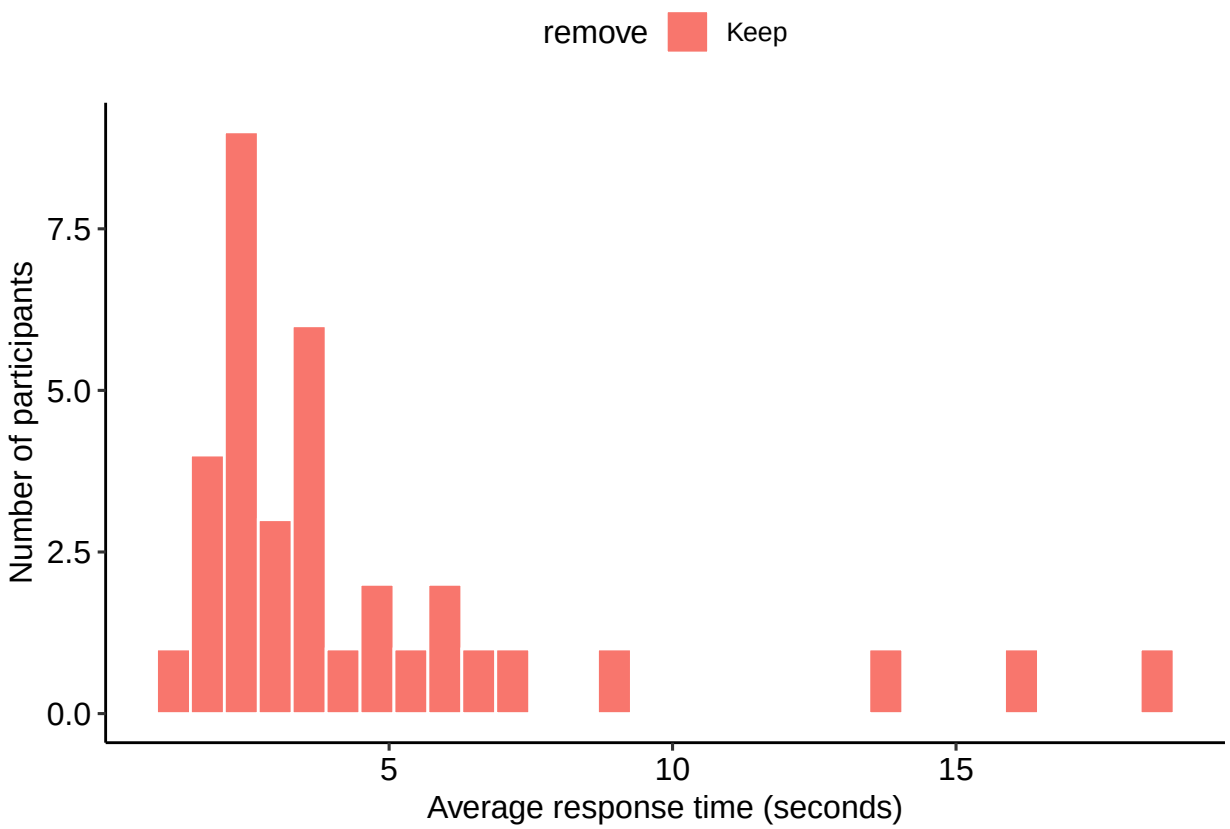


Figure 3: Distribution of average time to respond to personality items.

```
data_t1 = inner_join(data_t1, filter(timing_data, remove == "Keep")) %>%
  select(-remove)
```

Based on timing, we removed 0 participants.

We create a variable which indicates the Block 1 condition of each participant. This is used in two places: first, in recruiting participants at Time 2 (participants are given the same format at Time 2 as they received in Block 1), and second, in selecting the corret items during the test-retest analyses.

```
data_t1 = data_t1 %>%
  mutate(condition = case_when(
    !is.na(outgoing_a) ~ "A",
    !is.na(outgoing_b) ~ "B",
    !is.na(outgoing_c) ~ "C",
    !is.na(outgoing_d) ~ "D",
  ))
```

At this point, we'll extract the Prolific ID numbers. These participants will be eligible to take the survey at Time 2.

```
data_t1 %>%
  select(proid, condition) %>%
  write_csv(file = here("data/elligible_proid"))
```

1.4 Time 2

```
data_2A <- read_csv(here("deidentified data/data_time2_A.csv"))
data_2B <- read_csv(here("deidentified data/data_time2_B.csv"))
data_2C <- read_csv(here("deidentified data/data_time2_C.csv"))
data_2D <- read_csv(here("deidentified data/data_time2_D.csv"))
```

```
data_2 = data_2A %>%
  full_join(data_2B) %>%
  full_join(data_2C) %>%
  full_join(data_2D)
```

Rename the following columns.

```
data_2 = data_2 %>%
  rename(start_date2 = start_date,
         duration_in_seconds2 = duration_in_seconds)
```

We rename several columns, in order to facilitate the use of regular expressions later. Specifically, we remove the underscores (_) in the columns pertaining to broad-mindedness and self-disciplined.

```
names(data_2) = str_replace(names(data_2), "broad_mind", "broadmind")
names(data_2) = str_replace(names(data_2), "self_disciplind", "selfdisciplined")
```

We can also remove the meta-data (timing, etc) around two attention check adjectives, "human" and "asleep".

```
data_t1 = data_t1 %>%
  select(-starts_with("t_human"),
         -starts_with("t_asleep"))
```

1.4.1 Recode personality item responses to numeric

We recode the responses to personality items, which we downloaded as text strings. Here, all items end with `_3` and sometimes with `i`.

```
p_items_2 = str_extract(names(data_2), "^[[:alpha:]]*_[abcd]_3(i)?$")
p_items_2 = p_items_2[!is.na(p_items_2)]

personality_items_2 = select(data_2, proid, all_of(p_items_2))
```

We apply the recoding function to all personality items.

```
personality_items_2 = personality_items_2 %>%
  mutate(
    across(!c(proid), recode_p))
```

Now we merge this back into the data_2.

```
data_2 = select(data_2, -all_of(p_items_2))
data_2 = full_join(data_2, personality_items_2)
```

1.4.2 Drop bots and inattentive participants

This code recreates the steps outlined in detail above for Time 1. Please refer to the descriptions above for justification and explaination of the code presented here.

1.4.2.1 Based on ID

We also check that the ID in time 2 matches an ID in time 1.

```
data_2 = data_2 %>%
  filter(proid %in% data_t1$proid)
```

We removed 35 participants without valid Prolific IDs.

1.4.2.2 Based on patterns

We remove any participant who provides the same response to over half of the items (17 or more items) from a given block in a row. The distribution of runs in Time 2 is depicted in Figure 4.

```
# first, identify unique adjectives, in order
adjectives = p_items_2 %>%
  str_remove_all("_.") %>%
  unique()

# extract block 3 questions
block3 = data_2 %>%
```

```
  select(proid, all_of(p_items_2))

#rename variables
n = 0
for(i in adjectives){
  n = n+1
  names(block3) = str_replace(names(block3), i, paste0("trait", str_pad(n, 2, pad = "0")))
}


block3 = block3 %>%
  gather(item, response, -proid) %>%
  filter(!is.na(response)) %>%
  mutate(item = str_remove(item, "_3(i)?$")) %>%
  separate(item, into = c("item", "format")) %>%
  #select(-format) %>%
  spread(item, response)

block3_runs = numeric(length = nrow(block3))

for(i in 1:nrow(block3)){
  run = 0
  maxrun = 0
  for(j in 3:ncol(block3)){
    if(block3[i,j] == block3[i, j-1]){
      run = run+1
      if(run > maxrun) maxrun = run
      } else{ run = 0}
  }
  block3_runs[i] = maxrun
}

#add to data_2 frame
block3$block3_runs = block3_runs
```

```
#combine results
runs_data_2 = block3 %>%
  select(proid, block3_runs) %>%
  mutate(
    remove = case_when(
      block3_runs >= 17 ~ "Remove",
      TRUE ~ "Keep"
    ))
```

There were 0 participants who provided the same answer 17 or more times in a row. These participants were removed from the analyses.

```
data_2 = data_2 %>%
  full_join(select(runs_data_2, proid, remove)) %>%
  filter(remove != "Remove") %>%
  select(-remove)

rm(runs_data_2)
```

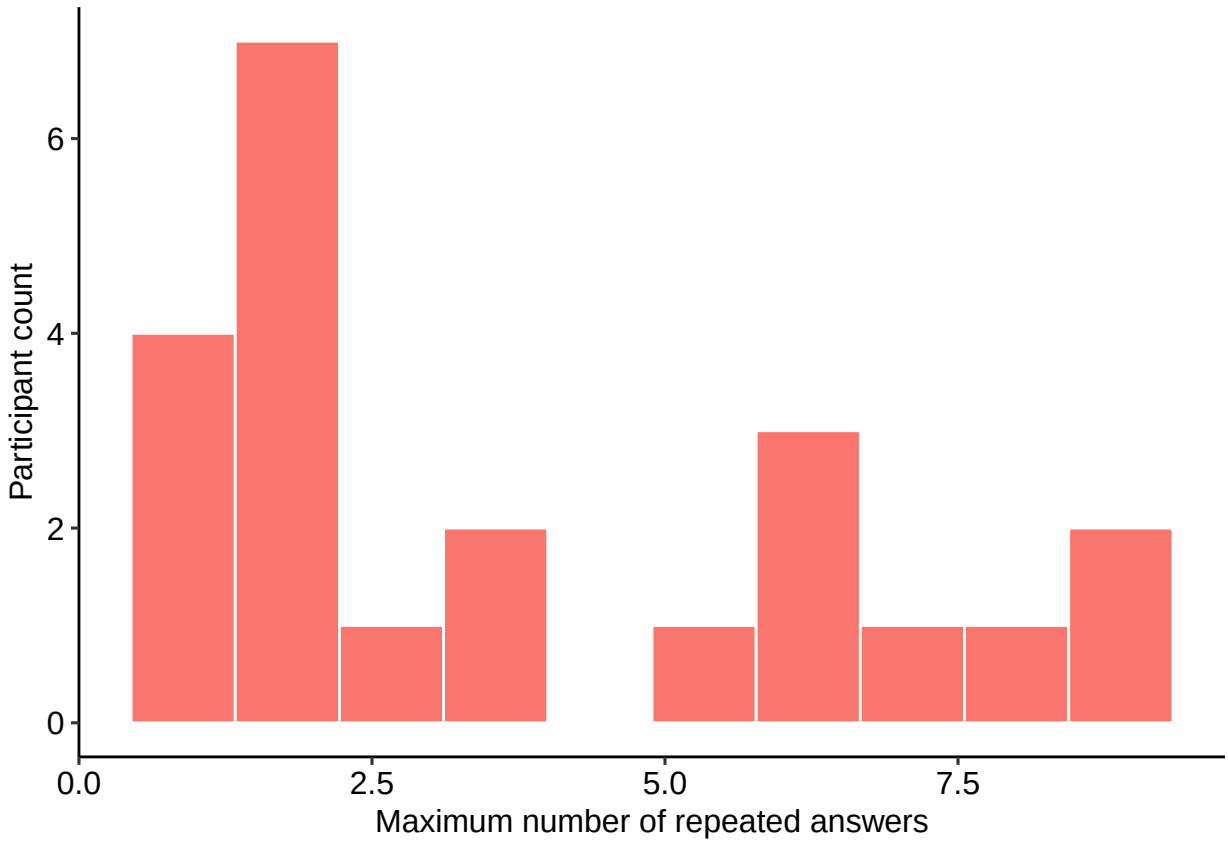


Figure 4: Maximum number of same consecutive responses in personality block 3.

1.4.2.3 Based on inattentive responding

Participants who respond positively to the adjective *asleep* or negatively to the word *human* are assumed to be inattentive. We filter out participants whose average response to these two items is greater than or equal to 4 (see Figure 5 for the distribution).

```
in_average = data_2 %>%
  # reverse score human
  mutate(across(matches("^human"),  ~(.x*-1)+7)) %>%
  # select id and attention check items
  select(proid, matches("^human"), matches("^asleep")) %>%
  gather(item, response, -proid) %>%
  filter(!is.na(response)) %>%
  group_by(proid) %>%
  summarise(avg = mean(response)) %>%
  mutate(
    remove = case_when(
      avg >= 4 ~ "Remove",
      TRUE ~ "Keep"))
```

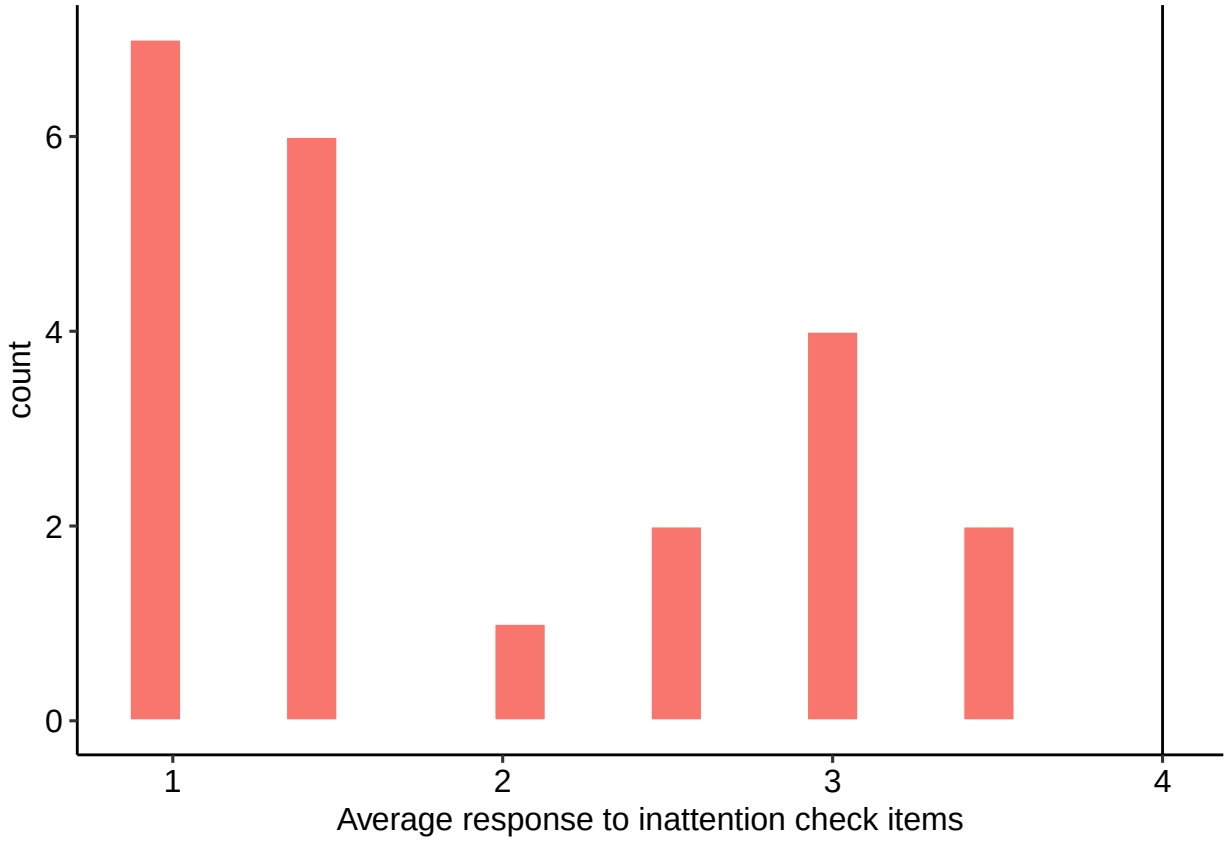


Figure 5: Average response to inattention check items

We remove 1 participants whose responses suggest inattention.

```
data_2 = data_2 %>%
  full_join(select(in_average, proid, remove)) %>%
  filter(remove != "Remove") %>%
  select(-remove)
```

1.4.2.4 Based on average time to respond to personality items

Participants who take too little (< 1 second) or too long (greater than 30 seconds) on average to answer each personality item are excluded. See Figure 6 for the distribution of average response time per item.

```
timing_data_2 = data_2 %>%
  select(proid, matches("t_[[:alpha:]]*_[abcd]_3(i)?_page_submit"))

timing_data_2 = timing_data_2 %>%
  gather(variable, timing, -proid) %>%
  filter(!is.na(timing))
```

To check, each participant should have the same number of responses: 33.

```
timing_data_2 %>%
  group_by(proid) %>%
  count() %>%
  ungroup() %>%
  summarise(min(n), max(n))
```

```
## # A tibble: 1 x 2
##   `min(n)` `max(n)`
##      <int>    <int>
## 1       33       33
```

```
timing_data_2 = timing_data_2 %>%
  group_by(proid) %>%
  summarise(m_time = mean(timing)) %>%
  mutate(remove = case_when(
    m_time < 1 ~ "Remove",
    m_time > 30 ~ "Remove",
    TRUE ~ "Keep"
  ))
```

```
data_2 = inner_join(data_2, filter(timing_data_2, remove == "Keep")) %>%
  select(-remove)
```

1.4.3 Merge all datasets together

We merge the Time 1 and Time 2 datasets together here.

```
data_2 = data_2 %>%
  select(proid, start_date2, duration_in_seconds2, very_delayed_recall, contains("_3")) %>%
  mutate(time2 = "yes") #indicates participant in time 2

data = data_t1 %>% full_join(data_2)
```

1.5 All data

1.5.1 Reverse score personality items

The following items are (typically) negatively correlated with the others: reckless, moody, worrying, nervous, careless, impulsive. We reverse-score them to ease interpretation of associations and means in the later

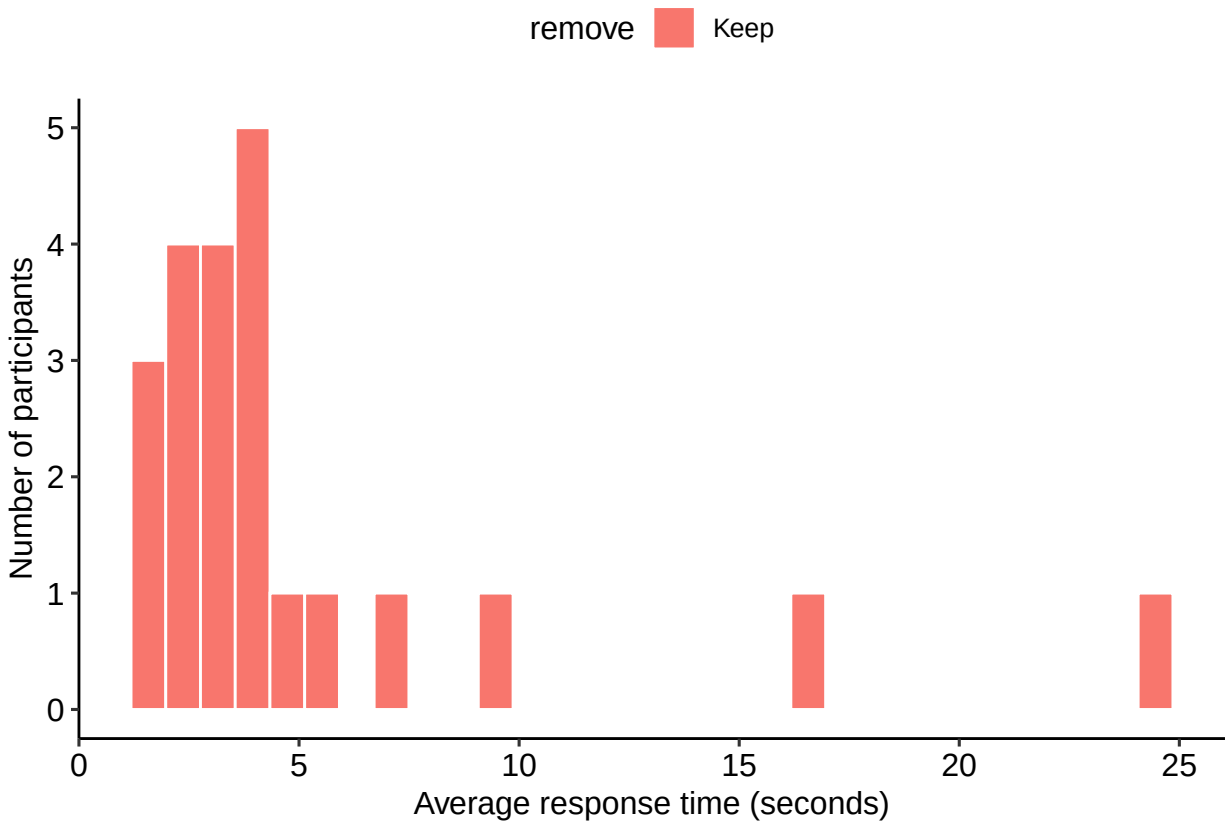


Figure 6: Distribution of average time to respond to personality items in Block 3.

sections. In short, all traits will be scored such that larger numbers are indicative of the more socially desirable end of the spectrum.

```
data = data %>%
  mutate(
    across(matches("^reckless"),  ~(.x*-1)+7),
    across(matches("^moody"),     ~(.x*-1)+7),
    across(matches("^worrying"),  ~(.x*-1)+7),
    across(matches("^nervous"),   ~(.x*-1)+7),
    across(matches("^careless"),  ~(.x*-1)+7),
    across(matches("^impulsive"), ~(.x*-1)+7))
```

We also create a vector noting the items that are reverse scored. We use this later in tables, to help identify patterns when looking at analyses within-adjective. We use this object elsewhere in the analyses.

```
reverse = c("reckless", "moody", "worrying", "nervous", "careless", "impulsive")
```

1.5.2 Score memory task

Now we score the memory task. We start by creating vectors of the correct responses.

```
correct1 = c("book", "child", "gold", "hotel", "king",
             "market", "paper", "river", "skin", "tree")

correct2 = c("butter", "college", "dollar", "earth", "flag",
             "home", "machine", "ocean", "sky", "wife")

correct3 = c("blood", "corner", "engine", "girl", "house",
             "letter", "rock", "shoes", "valley", "woman")

correct4 = c("baby", "church", "doctor", "fire", "garden",
             "palace", "sea", "table", "village", "water")
```

Next we convert all responses to lowercase. Then we break the string of responses into a vector containing many strings.

```
data = data %>%
  mutate(
    across(matches("recall"),tolower), # convert to lower
    #replace carriage return with space
    across(matches("recall"), str_replace_all, pattern = "\\n", replacement = ","),
    # remove spaces
    across(matches("recall"), str_replace_all, pattern = " ", replacement = ","),
    # remove doubles
    across(matches("recall"), str_replace_all, pattern = ",,", replacement = ","),
    #remove last comma
    across(matches("recall"), str_remove, pattern = ",$"),
    # split the strings based on the spaces
    across(matches("recall"), str_split, pattern = ","))
```

1.5.2.1 Immediate recall

Now we use the `amatch` function in the `stringdist` package to look for exact (or close) matches to the target words. This function returns for each word either the position of the key in which you can find the target word or `NA` to indicate the word or a close match does not exist in the string.

```
distance = 1 #maximum distance between target word and correct response
data = data %>%
  mutate(
    memory1 = map(recall1, ~sapply(., amatch, correct1, maxDist = distance)),
    memory2 = map(recall2, ~sapply(., amatch, correct2, maxDist = distance)),
    memory3 = map(recall3, ~sapply(., amatch, correct3, maxDist = distance)),
    memory4 = map(recall4, ~sapply(., amatch, correct4, maxDist = distance))
    )
```

We count the number of correct answers. This gets complicated; in lieu of writing out a paragraph explanation, we have opted for in-text comments to orient those interested in following the code.

```
data = data %>%
  mutate(
    across(starts_with("memory"),
      #replace position with 1
      ~map(., sapply, FUN = function(x) ifelse(x >0, 1, 0))),
    across(starts_with("recall"),
           # are there non-missing values in the original response?
           ~map_dbl(.,
                    .f = function(x) sum(!is.na(x))),
           .names = "{.col}_miss"),
    across(starts_with("memory"),
      #replace position with 1
      # count the number of correct answers
      ~map_dbl(., sum, na.rm=T))) %>%
  mutate(
    memory1 = case_when(
      # if there were no responses, make the answer NA
      recall1_miss == 0 ~ NA_real_,
      # otherwise, the number of correct guesses
      TRUE ~ memory1),
    memory2 = case_when(
      recall2_miss == 0 ~ NA_real_,
      TRUE ~ memory2),
    memory3 = case_when(
      recall3_miss == 0 ~ NA_real_,
      TRUE ~ memory3),
    memory4 = case_when(
      recall4_miss == 0 ~ NA_real_,
      TRUE ~ memory4)) %>%
  # no longer need the missing count variables
  select(-ends_with("miss"))
```

Finally, we want to go from 4 columns (one for each recall test), to two: one that has the number of correct responses, and one that indicates which version they saw.

```
data = data %>%
  select(proid, starts_with("memory")) %>%
```

```
  gather(mem_condition, memory, -proid) %>%
  filter(!is.na(memory)) %>%
  mutate(mem_condition = str_remove(mem_condition, "memory")) %>%
  full_join(data)
```

To demonstrate the accuracy of the code, here we present a random subset of participants' raw responses and their assigned memory score.

```
#from memory condition 1
data %>%
  filter(mem_condition == 1) %>%
  select(recall1, memory) %>%
  sample_n(3) %>%
  mutate(recall1 = map_chr(recall1, paste, collapse = ", "))
```

```
## # A tibble: 3 x 2
##   recall1                                                        memory
##   <chr>                                                           <dbl>
## 1 book, hotel, market, chain, paper, gold                             5
## 2 tree, skin, river, paper, market, king, hotel, gold, child, book   10
## 3 book, child, gold, hotel, river, tree, market                       7
```

```
#from memory condition 2
data %>%
  filter(mem_condition == 2) %>%
  select(recall2, memory) %>%
  sample_n(3) %>%
  mutate(recall2 = map_chr(recall2, paste, collapse = ", "))
```

```
## # A tibble: 3 x 2
##   recall2                                                      memory
##   <chr>                                                         <dbl>
## 1 college                                                           1
## 2 machine, wife, health                                             2
## 3 butter, college, dollar, earth, flag, home, machine, ocean, sky, wife   10
```

```
#from memory condition 3
data %>%
  filter(mem_condition == 3) %>%
  select(recall3, memory) %>%
  sample_n(3) %>%
  mutate(recall3 = map_chr(recall3, paste, collapse = ", "))
```

```
## # A tibble: 3 x 2
##   recall3                                                        memory
##   <chr>                                                           <dbl>
## 1 block, corner, engine, girl, house, letter, woman, shoes            7
## 2 girl, woman, valley, shoes, rock, house, engine                     7
## 3 blood, corner, engine, girl, house, letter, rock, shoes, valley, woman.   10
```

Table 1: Memory responses by condition

Condition	Mean	SD	Min	Max	N
1	5.50	2.56	1	10	8
2	4.62	3.20	1	10	8
3	6.40	2.72	1	10	10
4	6.44	2.51	2	10	9

```
#from memory condition 4
data %>%
  filter(mem_condition == 4) %>%
  select(recall4, memory) %>%
  sample_n(3) %>%
  mutate(recall4 = map_chr(recall4, paste, collapse = ", "))
```

```
## # A tibble: 3 x 2
##   recall4                                                                memory
##   <chr>                                                                   <dbl>
## 1 sea, table, water, doctor, village, baby, fire, church, garden, palace     10
## 2 baby, church, fire, garden, palace, sea, table, water                       8
## 3 baby, church, water, fire, garden                                           5
```

Participants remember on average 5.80 words correctly ($SD = 2.73$).

1.5.2.2 Delayed recall A challenge with the delayed recall task is identifying the memory condition that participants were assigned to, but this is made easier by the work done above. The following code mainly reproduces the steps used for scoring the immediate memory recall task. The main difference is that we have a single column containing all responses (`delayed_recall`), regardless of which memory condition participants were assigned to. We score this response against all four answer keys, then select the maximum (best) score.

```
mem2 = data %>%
  select(proid, mem_condition, delayed_recall) %>%
  mutate(newid = 1:nrow(.))

mem2 = mem2 %>%
  mutate(
    delayed_recall1 = map(delayed_recall, ~sapply(., amatch, correct1, maxDist = distance)),
    delayed_recall2 = map(delayed_recall, ~sapply(., amatch, correct2, maxDist = distance)),
    delayed_recall3 = map(delayed_recall, ~sapply(., amatch, correct3, maxDist = distance)),
    delayed_recall4 = map(delayed_recall, ~sapply(., amatch, correct4, maxDist = distance))
    ) %>%
  gather(variable, delayed_memory, delayed_recall1:delayed_recall4)

mem2 = mem2 %>%
  mutate(
      delayed_memory = map(delayed_memory, sapply,
                  FUN = function(x) ifelse(x >0, 1, 0)),
      # count the number of correct answers
      delayed_memory = map_dbl(delayed_memory, sum, na.rm=T))
```

```
mem2 = mem2 %>%
  group_by(proid) %>%
  filter(delayed_memory == max(delayed_memory)) %>%
  filter(row_number() == 1 ) %>%
  select(-delayed_recall, -variable, -newid)

data = inner_join(data, mem2)
```

Participants remember on average 5.11 words correctly after 5-10 minutes ($SD = 2.97$).

1.5.2.3 Very-delayed recall Finally, we score the memory challenge posed at Time 2. Like scoring the delayed recall task, we have a single column containing responses fromo all participants, regardless of the original memory condition.

```
mem3 = data %>%
  filter(time2 == "yes") %>%
  select(proid, mem_condition, very_delayed_recall) %>%
  mutate(newid = 1:nrow(.))

mem3 = mem3 %>%
  mutate(
    very_delayed_recall1 = map(very_delayed_recall, ~sapply(., amatch, correct1, maxDist = distance)),
    very_delayed_recall2 = map(very_delayed_recall, ~sapply(., amatch, correct2, maxDist = distance)),
    very_delayed_recall3 = map(very_delayed_recall, ~sapply(., amatch, correct3, maxDist = distance)),
    very_delayed_recall4 = map(very_delayed_recall, ~sapply(., amatch, correct4, maxDist = distance))
    ) %>%
  gather(variable, very_delayed_memory, very_delayed_recall1:very_delayed_recall4)

mem3 = mem3 %>%
  mutate(
      very_delayed_memory = map(very_delayed_memory, sapply,
                  FUN = function(x) ifelse(x >0, 1, 0)),
      # count the number of correct answers
      very_delayed_memory = map_dbl(very_delayed_memory, sum, na.rm=T))

mem3 = mem3 %>%
  group_by(proid) %>%
  filter(very_delayed_memory == max(very_delayed_memory)) %>%
  filter(row_number() == 1 ) %>%
  select(-very_delayed_recall, -variable, -newid)

data = full_join(data, mem3)
```

1.5.2.4 Correlations Figure 7 displays the univariate and bivariate distributions of the memory scores and the bivariate correlations. In general, there was good spread in the immediate recall and delayed (10 minute) recall variables. Few participants remembered any of the words after two weeks.

```
data %>%
  select(matches("memory$")) %>%
  corr.test
```

```
## Call:corr.test(x = .)
## Correlation matrix
##                       memory delayed_memory very_delayed_memory
## memory                  1.00           0.84                0.07
## delayed_memory          0.84           1.00                0.06
## very_delayed_memory     0.07           0.06                1.00
## Sample Size
##                       memory delayed_memory very_delayed_memory
## memory                    35             35                  22
## delayed_memory            35             35                  22
## very_delayed_memory       22             22                  22
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##                       memory delayed_memory very_delayed_memory
## memory                  0.00           0.00                   1
## delayed_memory          0.00           0.00                   1
## very_delayed_memory     0.76           0.81                   0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

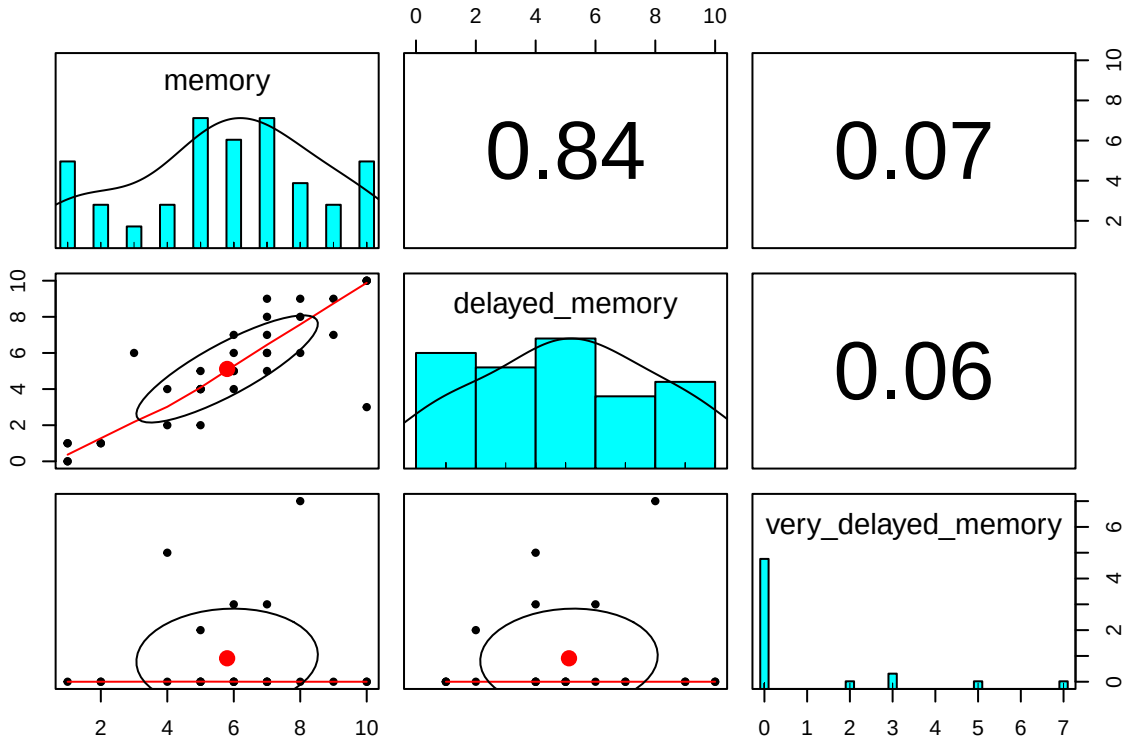


Figure 7: Distributions of memory scores across booth time points.

1.5.3 Change labels of device variable

Longer labels were provided to participants for clarity. However, we will use shorter labels in our analyses and figures.

```
data = data %>%
  mutate(devicetype = factor(
```

```
    devicetype,
    levels = c("Desktop or laptop computer", "Mobile",
               "Tablet (for example, iPad, Galaxy Tablet, Amazon Fire, etc.)"),
    labels = c("Computer", "Mobile", "Tablet")
  ))
```

1.5.4 Reorder demographic categories

We set the order of ordinal demographic variables, which helps generate more interpretable figures and tables.

```
data = data %>%
  mutate(edu = factor(edu,
                      levels = c(
                        "Less than 12 years",
                        "High school graduate/GED",
                        "Currently in college/university",
                        "Some college/university, but did not graduate",
                        "Associate degree (2 year)",
                        "College/university degree (4 year)",
                        "Currently in graduate or professional school",
                        "Graduate or professional school degree"))) %>%
  mutate(hhinc = str_remove(hhinc, " a year"),
         hhinc = str_replace_all(hhinc, ",000", "K"),
         hhinc = str_replace_all(hhinc, " to ", "-"),
         hhinc = str_replace_all(hhinc, "less than", "<"),
         hhinc = str_replace_all(hhinc, "more than", ">"))%>%
  mutate(hhinc = factor(hhinc,
                        levels = c(
                          "< $20,000",
                          "$20K-$40K",
                          "$40K-$60K",
                          "$60K-$80K",
                          "$80K-$100K",
                          "$100K-$120K",
                          "$120K-$150K",
                          "$150K-$200K",
                          "$200K-$250K",
                          "$250K-$350K",
                          "$350K-$500K",
                          ">$500K"
                        )))
```

1.5.5 Long-form dataset

We need one dataset that contains the responses to and timing of the personality items in long form. This will be used for nearly all the statistical models, which will nest items within person. To create this, we first select the responses to the items of different formats. For this set of analyses, we use data collected in both Block 1 and Block 2 – that is, each participant saw the same format for every item during Block 1, but a random format for each item in Block 2.

These variable names have one of four formats: `[trait]_[abcd]` (for example, `talkative_a`),

`[trait]_[abcd]_2` (for example, `talkative_a_2`), `[trait]_[abcd]_3` (e.g., `talkative_a_3`), or `[trait]_[abcd]_3i` (e.g., `talkative_a_3i`). We search for these items using regular expressions.

```
item_responses = str_subset(
  names(data),
  "^([[:alpha:]])+_[abcd](_2)?(_3)?(i)?$"
)
```

Similarly, we'll need to know how long it took participants to respond to these items. These variable names have one of four formats listed above followed by the string `page_submit`. We search for these items using regular expressions.

```
item_timing = str_subset(names(data), "t_([[:alpha:]])+_[abcd](_2)?(_3)?(i)?_page_submit$")
```

We extract just the participant IDs, delayed memory, and these variables.

```
items_df = data %>%
  select(proid, condition, time2,
         memory, delayed_memory, very_delayed_memory,
         devicetype,
         all_of(item_responses), all_of(item_timing))
```

Next we reshape these data into long form. This requires several steps. We'll need to identify whether each value is a response or timing; we can use the presence of the string `t_` for this. Next, we'll identify the block based on whether the string contains `_2` or `_3`. We also identify whether it ends with `i`, indicating the item in block 3 started with "I". Then, we identify the condition based on which letter (`a`, `b`, `c`, or `d`) follows an underscore. Throughout, we'll strip the item string of extraneous information until we're left with only the adjective assessed. Finally, we'll use spread to create separate columns for the response and the timing variables.

```
items_df = items_df %>%
  gather(item, value, all_of(item_responses), all_of(item_timing)) %>%
  filter(!is.na(value)) %>%
  # identify whether timing or response
  mutate(variable = ifelse(str_detect(item, "^t_"), "timing", "response"),
         item = str_remove(item, "^t_"),
         item = str_remove(item, "_page_submit$")) %>%
  #identify block
  mutate(
    block = case_when(
      str_detect(item, "_2") ~ "2",
      str_detect(item, "_3") ~ "3",
      TRUE ~ "1"),
    item = str_remove(item, "_[23]")) %>%
  # identify presence of "I"
  mutate(i = case_when(
    str_detect(item, "i$") ~ "Present",
    TRUE ~ "Absent"),
    item = str_remove(item, "i$")) %>%
  separate(item, into = c("item", "format")) %>%
  spread(variable, value)
```

1.5.5.1 Remove 'human' and 'asleep'

We also remove responses to the adjectives "human" and "asleep", as these are not personality items per-se and included for the purpose of attention checks.

```
items_df = items_df %>%
  filter(item != "human") %>%
  filter(item != "asleep")
```

1.5.5.2 Label formatting conditions

We give labels to the formats, to clarify interpretations and aid table and figure construction.

```
items_df$format = as.factor(items_df$format)
items_df$format = relevel(items_df$format, ref = "a")
items_df$format = factor(items_df$format,
                                levels = c("a","b","c","d"),
                                labels = c("Adjective\nOnly", "Am\nAdjective", "Tend to be\nAdjective",
```

1.5.5.3 Transform seconds

The variable `seconds` appears to have a very severe right skew (see Figure 8). We log-transform this variable for later analyses.

```
items_df = items_df %>%
  mutate(seconds_log = log(timing))
```

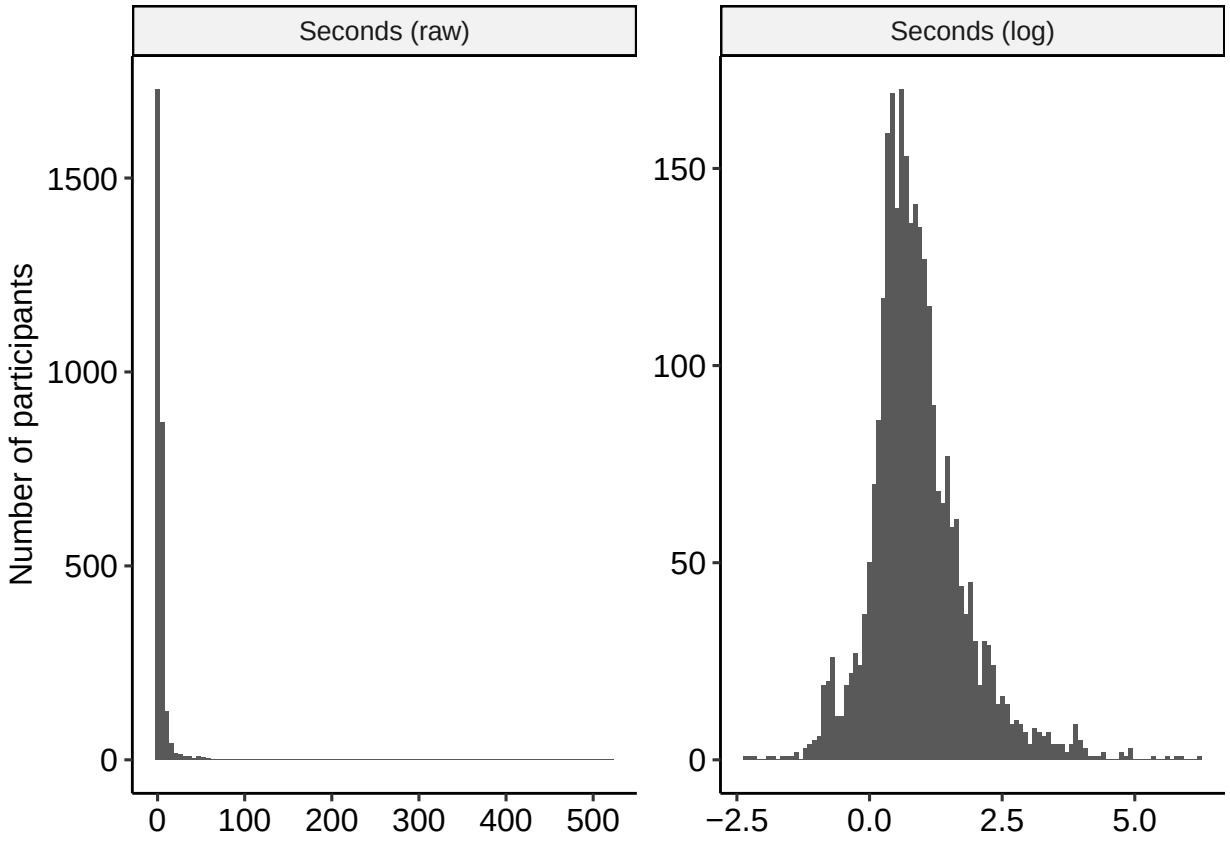


Figure 8: Distribution of seconds, raw and transformed.

2 Descriptives

Participants (N = 35; 22.86% female) were, on average, 39.40 years old (SD = 6.02, minimum = 31, maximum = 51; see Figure 9A for the full distribution). A majority (65.71%) of participants identified as White only (25.71% only); Figure 9B shows the other response options and frequencies. See Figure 9C for the distribution of education, and 9D for the distribution of household income.

2.1 Time

How much time elapsed between assessments?

```
data = data %>%
  mutate(difference = as.numeric(start_date2-start_date))
summary(data$difference)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   11.96   12.00   12.49   13.26   14.17   17.48      13
```

How long did it take participants to complete the Time 1 survey?

```
summary(data$duration_in_seconds/60)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.65   13.98   17.55   20.56   25.08   44.27
```

How long did it take participants to complete the Time 2 survey?

```
summary(data$duration_in_seconds2/60)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   2.000   3.329   4.925   8.910   7.875  40.833      13
```

2.2 Personality by block and format

See Table 2 for the descriptive statistics of each format by block.

See Table 3 for the descriptive statistics of each item and format in Block 1 (Time 1).

See Table 4 for the descriptive statistics of each item and format in Block 2 (Time 1).

See Table 5 for the descriptive statistics of each item and format in Block 3 (Time 2).

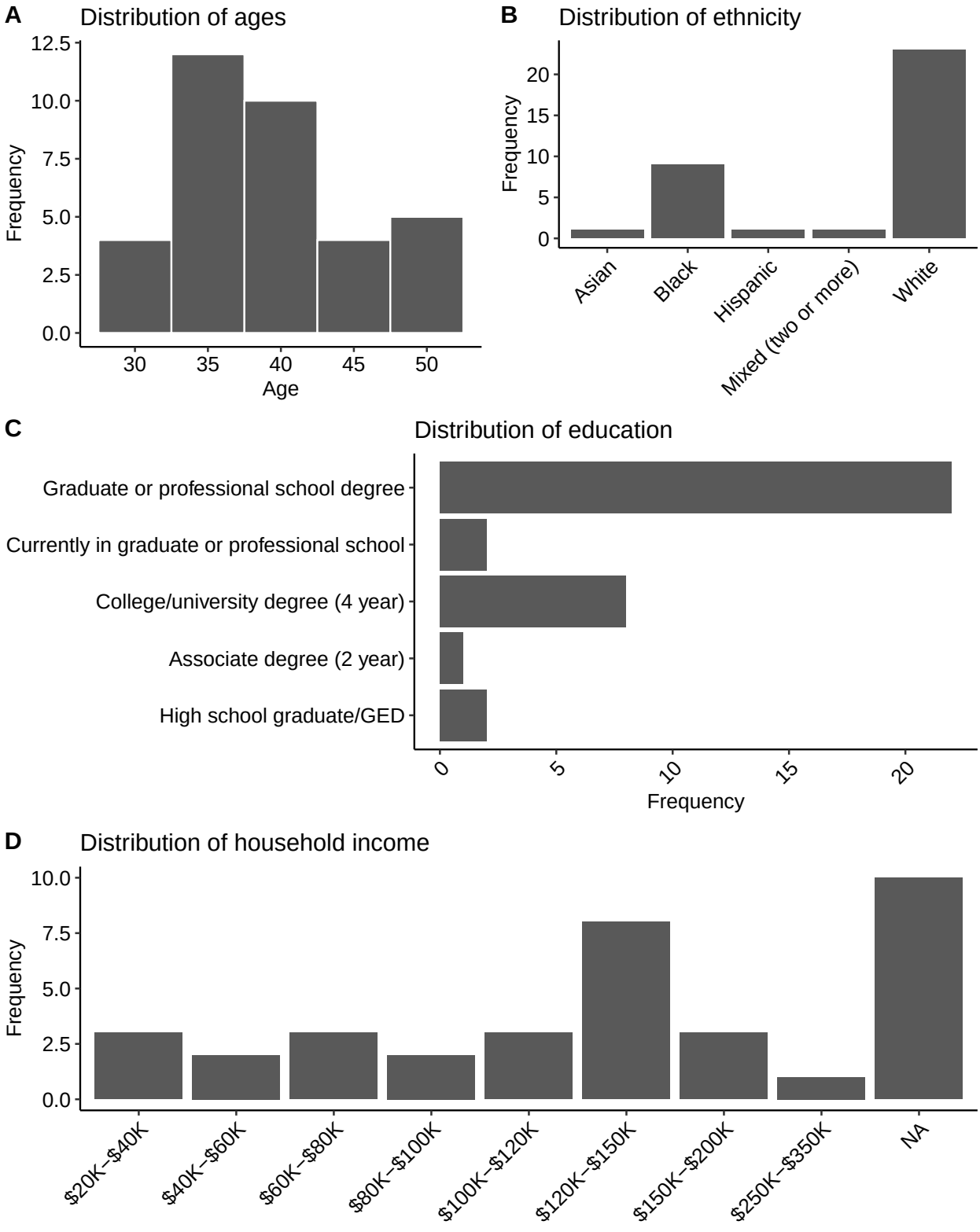


Figure 9: Distributions of key demographics across the entire sample

Table 2: Descriptives of responses by format and block

Block	Format	M	SD	Median	N (responses)	N (participants)
1	Adjective Only	4.70	1	5	341	11
1	Am Adjective	4.52	1	5	279	9
1	Tend to be Adjective	4.63	1	5	248	8
1	Am someone who tends to be Adjective	4.73	1	5	217	7
2	Adjective Only	4.68	1	5	273	35
2	Am Adjective	4.57	1	5	279	35
2	Tend to be Adjective	4.67	1	5	269	35
2	Am someone who tends to be Adjective	4.61	1	5	264	35
3	Adjective Only	4.87	1	5	186	6
3	Am Adjective	4.67	1	5	217	7
3	Tend to be Adjective	4.46	1	5	186	6
3	Am someone who tends to be Adjective	5.06	1	6	93	3

3 Does item format affect response?

The primary aims of this study are to evaluate the effects of item wording in online, self-report personality assessment. Specifically, we intend to consider the extent to which incremental wording changes may influence differences in the distributions of responses, response times, and psychometric properties of the items. These wording changes will include a progression from using (1) trait-descriptive adjectives by themselves, (2) with the linking verb "to be" (Am...), (3) with the additional verb "to tend" (Tend to be...), and (4) with the pronoun "someone" (Am someone who tends to be...).

Using a protocol that administers each adjective twice to the same participant (in different combinations of item format administered randomly across participants), we will use between-person analyses to compare responses using group-level data for the different formats.

These analyses will attempt to account for delayed_memory effects by collecting data on immediate and delayed recall (5 minutes and approximately two weeks) using a delayed_memory paradigm that was developed based on a similar recall task used in the HRS (Runge et al., 2015).

3.1 Effect of format (Block 1 data)

We used a multilevel model, nesting response within participant to account for dependence. Our primary predictor was format. Here, we use only Block 1 data; in other words, effects are largely between person, although each person contributes 31 unique data points to the analysis (one for each trait). We use the `anova` function to estimate the amount of variability in response due to format.

```
item_block1 = filter(items_df, block == "1")

mod.format_b1 = lmer(response~format + (1|proid),
                  data = item_block1)
anova(mod.format_b1)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF DenDF F value Pr(>F)
## format 1.3559 0.45198     3    31  0.3033 0.8228
```

Table 3: Descriptives of responses to Block 1 by format and item

item	Adjective Only	Am Adjective	Tend to be Adjective	Am someone who tends to be Adjective
active	5.45 (1.21)	5.00 (0.87)	4.75 (1.04)	4.86 (1.77)
adventurous	4.82 (0.60)	5.00 (0.87)	4.50 (0.93)	4.57 (1.40)
broadminded	4.55 (1.29)	5.11 (1.05)	5.00 (0.93)	4.57 (0.79)
calm	5.45 (0.52)	4.67 (1.00)	4.75 (1.28)	5.00 (1.15)
careless	4.82 (1.40)	3.33 (1.94)	4.12 (1.25)	5.29 (1.11)
caring	5.36 (0.67)	4.78 (0.83)	5.00 (1.07)	5.43 (0.79)
cautious	4.91 (1.14)	4.67 (1.50)	5.00 (0.53)	4.43 (1.72)
creative	5.09 (0.94)	4.67 (1.22)	5.00 (0.93)	5.43 (0.79)
curious	4.64 (1.12)	4.89 (0.78)	4.62 (1.30)	4.57 (0.98)
friendly	5.55 (0.69)	5.33 (0.71)	5.25 (0.71)	5.29 (0.76)
hardworking	5.45 (0.69)	5.44 (0.73)	5.38 (1.06)	5.43 (0.53)
helpful	5.09 (0.94)	5.22 (0.83)	5.12 (0.64)	5.29 (1.11)
imaginative	5.00 (1.00)	5.22 (0.97)	5.25 (0.89)	5.57 (0.53)
impulsive	3.00 (1.18)	3.11 (1.17)	3.00 (1.51)	3.71 (1.80)
intelligent	5.09 (1.22)	4.78 (1.64)	5.25 (0.89)	5.43 (0.53)
lively	4.91 (0.83)	4.56 (0.73)	5.00 (1.20)	5.00 (1.41)
moody	3.91 (1.30)	2.89 (1.36)	3.38 (1.19)	3.86 (1.57)
nervous	4.09 (1.70)	3.56 (1.59)	3.88 (0.99)	4.00 (1.91)
organized	5.27 (0.79)	4.67 (1.12)	4.75 (1.28)	4.86 (1.07)
outgoing	4.91 (0.83)	4.89 (1.05)	4.38 (1.19)	4.71 (1.60)
reckless	4.18 (1.72)	4.11 (1.83)	4.38 (1.69)	5.14 (0.90)
responsible	5.64 (0.50)	5.22 (0.83)	5.50 (0.76)	5.43 (0.53)
selfdisciplined	4.91 (1.76)	5.11 (0.60)	4.75 (0.71)	5.57 (0.79)
softhearted	5.18 (0.98)	4.78 (0.97)	4.88 (0.99)	4.71 (1.80)
sophisticated	3.73 (1.27)	4.11 (1.05)	4.12 (1.73)	3.86 (1.07)
sympathetic	5.27 (1.01)	4.56 (1.01)	5.00 (0.76)	4.57 (1.27)
talkative	2.73 (1.01)	4.22 (1.72)	4.12 (1.46)	2.71 (1.38)
thorough	4.36 (1.29)	4.56 (1.13)	4.38 (1.06)	4.57 (0.98)
thrifty	3.64 (1.12)	3.89 (1.27)	4.75 (1.28)	3.43 (0.79)
warm	5.00 (1.48)	4.78 (0.83)	5.12 (0.99)	5.14 (0.69)
worrying	3.64 (1.43)	2.89 (1.69)	3.12 (1.25)	4.29 (1.80)

Table 4: Descriptives of responses to Block 2 by format and item

item	Adjective Only	Am Adjective	Tend to be Adjective	Am someone who tends to be Adjective
active	4.56 (1.67)	5.22 (1.30)	4.89 (1.27)	5.12 (0.35)
adventurous	5.00 (1.05)	4.86 (0.90)	5.22 (0.67)	4.11 (1.36)
broadminded	5.29 (0.49)	4.70 (0.95)	5.22 (0.67)	4.78 (0.67)
calm	5.10 (0.57)	5.25 (0.71)	4.88 (0.83)	4.89 (1.05)
careless	4.50 (1.43)	3.62 (2.13)	4.62 (1.51)	4.89 (1.05)
caring	5.11 (0.60)	5.50 (0.76)	4.75 (0.46)	5.20 (0.79)
cautious	4.67 (0.71)	4.80 (0.92)	5.00 (0.53)	4.62 (1.30)
creative	5.00 (0.58)	5.50 (0.71)	5.10 (0.57)	4.88 (1.36)
curious	4.25 (1.58)	4.30 (1.64)	4.67 (1.32)	5.38 (0.74)
friendly	5.25 (0.71)	5.00 (1.31)	5.33 (0.71)	5.00 (0.67)
hardworking	5.44 (0.73)	4.60 (1.17)	6.00 (0.00)	5.12 (0.35)
helpful	5.33 (0.50)	5.40 (0.70)	5.43 (0.53)	5.56 (0.73)
imaginative	4.70 (0.67)	5.22 (0.44)	5.12 (1.13)	5.25 (0.71)
impulsive	3.89 (1.62)	3.22 (1.79)	3.75 (1.67)	2.89 (0.93)
intelligent	5.12 (0.64)	5.00 (1.32)	5.44 (0.53)	5.33 (0.71)
lively	4.75 (1.04)	5.00 (1.00)	5.00 (1.00)	4.00 (1.94)
moody	3.73 (0.79)	4.00 (1.94)	3.00 (1.91)	4.00 (1.69)
nervous	4.12 (1.46)	4.10 (1.52)	3.11 (1.83)	3.75 (1.67)
organized	4.62 (1.51)	4.70 (1.42)	5.50 (0.76)	4.89 (0.60)
outgoing	5.12 (0.99)	4.40 (1.35)	4.12 (1.55)	4.78 (0.97)
reckless	4.56 (1.33)	4.89 (1.54)	3.80 (1.87)	4.86 (1.46)
responsible	5.30 (0.48)	5.22 (0.83)	5.00 (0.58)	4.78 (1.72)
selfdisciplined	5.00 (0.93)	4.67 (1.41)	4.80 (1.23)	5.00 (0.76)
softhearted	4.90 (0.88)	4.78 (1.30)	5.62 (0.52)	5.00 (1.07)
sophisticated	3.89 (1.17)	4.14 (1.68)	3.40 (1.35)	4.33 (0.87)
sympathetic	5.00 (0.87)	4.78 (0.83)	4.22 (0.97)	5.00 (0.76)
talkative	3.67 (1.00)	2.22 (1.20)	3.88 (1.81)	3.11 (1.83)
thorough	4.62 (1.19)	4.89 (1.05)	4.30 (1.25)	4.88 (0.83)
thrifty	4.00 (1.41)	3.78 (1.20)	4.33 (1.12)	3.56 (1.51)
warm	5.33 (0.50)	5.00 (0.94)	5.11 (0.60)	5.29 (0.76)
worrying	3.56 (1.24)	2.43 (1.62)	4.45 (1.69)	2.88 (1.64)

Table 5: Descriptives of items to Block 3 by format

item	Adjective Only	Am Adjective	Tend to be Adjective	Am someone who tends to be Adjective
active	4.56 (1.67)	5.22 (1.30)	4.89 (1.27)	5.12 (0.35)
adventurous	5.00 (1.05)	4.86 (0.90)	5.22 (0.67)	4.11 (1.36)
broadminded	5.29 (0.49)	4.70 (0.95)	5.22 (0.67)	4.78 (0.67)
calm	5.10 (0.57)	5.25 (0.71)	4.88 (0.83)	4.89 (1.05)
careless	4.50 (1.43)	3.62 (2.13)	4.62 (1.51)	4.89 (1.05)
caring	5.11 (0.60)	5.50 (0.76)	4.75 (0.46)	5.20 (0.79)
cautious	4.67 (0.71)	4.80 (0.92)	5.00 (0.53)	4.62 (1.30)
creative	5.00 (0.58)	5.50 (0.71)	5.10 (0.57)	4.88 (1.36)
curious	4.25 (1.58)	4.30 (1.64)	4.67 (1.32)	5.38 (0.74)
friendly	5.25 (0.71)	5.00 (1.31)	5.33 (0.71)	5.00 (0.67)
hardworking	5.44 (0.73)	4.60 (1.17)	6.00 (0.00)	5.12 (0.35)
helpful	5.33 (0.50)	5.40 (0.70)	5.43 (0.53)	5.56 (0.73)
imaginative	4.70 (0.67)	5.22 (0.44)	5.12 (1.13)	5.25 (0.71)
impulsive	3.89 (1.62)	3.22 (1.79)	3.75 (1.67)	2.89 (0.93)
intelligent	5.12 (0.64)	5.00 (1.32)	5.44 (0.53)	5.33 (0.71)
lively	4.75 (1.04)	5.00 (1.00)	5.00 (1.00)	4.00 (1.94)
moody	3.73 (0.79)	4.00 (1.94)	3.00 (1.91)	4.00 (1.69)
nervous	4.12 (1.46)	4.10 (1.52)	3.11 (1.83)	3.75 (1.67)
organized	4.62 (1.51)	4.70 (1.42)	5.50 (0.76)	4.89 (0.60)
outgoing	5.12 (0.99)	4.40 (1.35)	4.12 (1.55)	4.78 (0.97)
reckless	4.56 (1.33)	4.89 (1.54)	3.80 (1.87)	4.86 (1.46)
responsible	5.30 (0.48)	5.22 (0.83)	5.00 (0.58)	4.78 (1.72)
selfdisciplined	5.00 (0.93)	4.67 (1.41)	4.80 (1.23)	5.00 (0.76)
softhearted	4.90 (0.88)	4.78 (1.30)	5.62 (0.52)	5.00 (1.07)
sophisticated	3.89 (1.17)	4.14 (1.68)	3.40 (1.35)	4.33 (0.87)
sympathetic	5.00 (0.87)	4.78 (0.83)	4.22 (0.97)	5.00 (0.76)
talkative	3.67 (1.00)	2.22 (1.20)	3.88 (1.81)	3.11 (1.83)
thorough	4.62 (1.19)	4.89 (1.05)	4.30 (1.25)	4.88 (0.83)
thrifty	4.00 (1.41)	3.78 (1.20)	4.33 (1.12)	3.56 (1.51)
warm	5.33 (0.50)	5.00 (0.94)	5.11 (0.60)	5.29 (0.76)
worrying	3.56 (1.24)	2.43 (1.62)	4.45 (1.69)	2.88 (1.64)

When examining only Block 1 data, item format was unassociated with participants' responses to personality items ($F(3, 31.00) = 0.30, p = .823$). See Figure 10 for a visualization of this effect. In addition, Figure 11 shows the full distribution of responses across format.

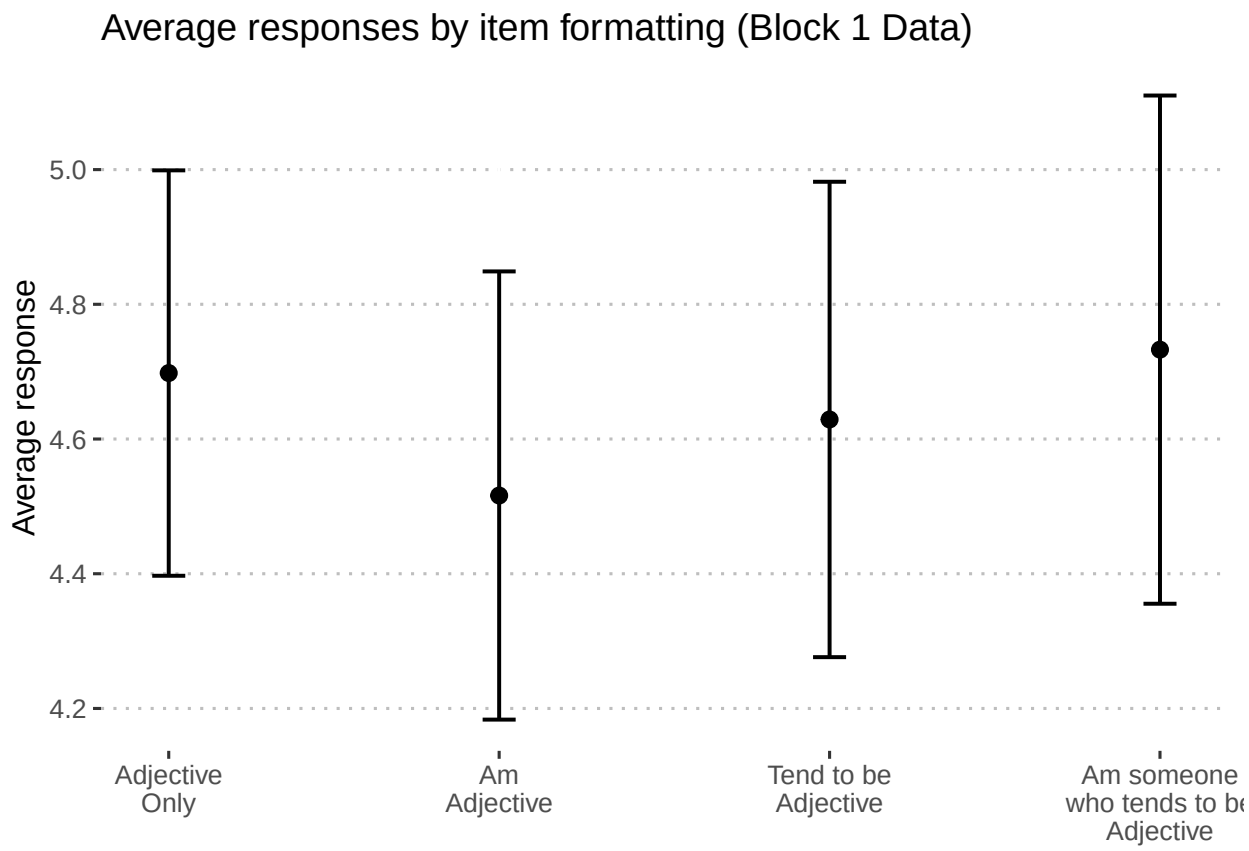


Figure 10: Predicted response on personality items by condition, using only Block 1 data.

3.1.1 One model for each adjective

We repeat this analysis separately for each trait. Because there is only one response per participant (when using only Block 1 data), we can drop the use of multilevel models and instead rely on a simple general linear model to test our hypothesis. Sepecfically, we test whether the proportion of variance attributable to item format is statistically significant.

```
mod_by_item_b1 = item_block1 %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~lm(response~format, data = .))) %>%
  mutate(aov = map(mod, anova))
```

We apply a Holm correction to the p-values extracted from these analyses, to adjust for the number of tests conducted. We present results in Table @ref(tab:mod1_item), which is organized by whether items were reverse-coded prior to analysis.

\begin{table}

\caption{(#tab:mod1_item)Format effects on response by item (block 1 data only)}

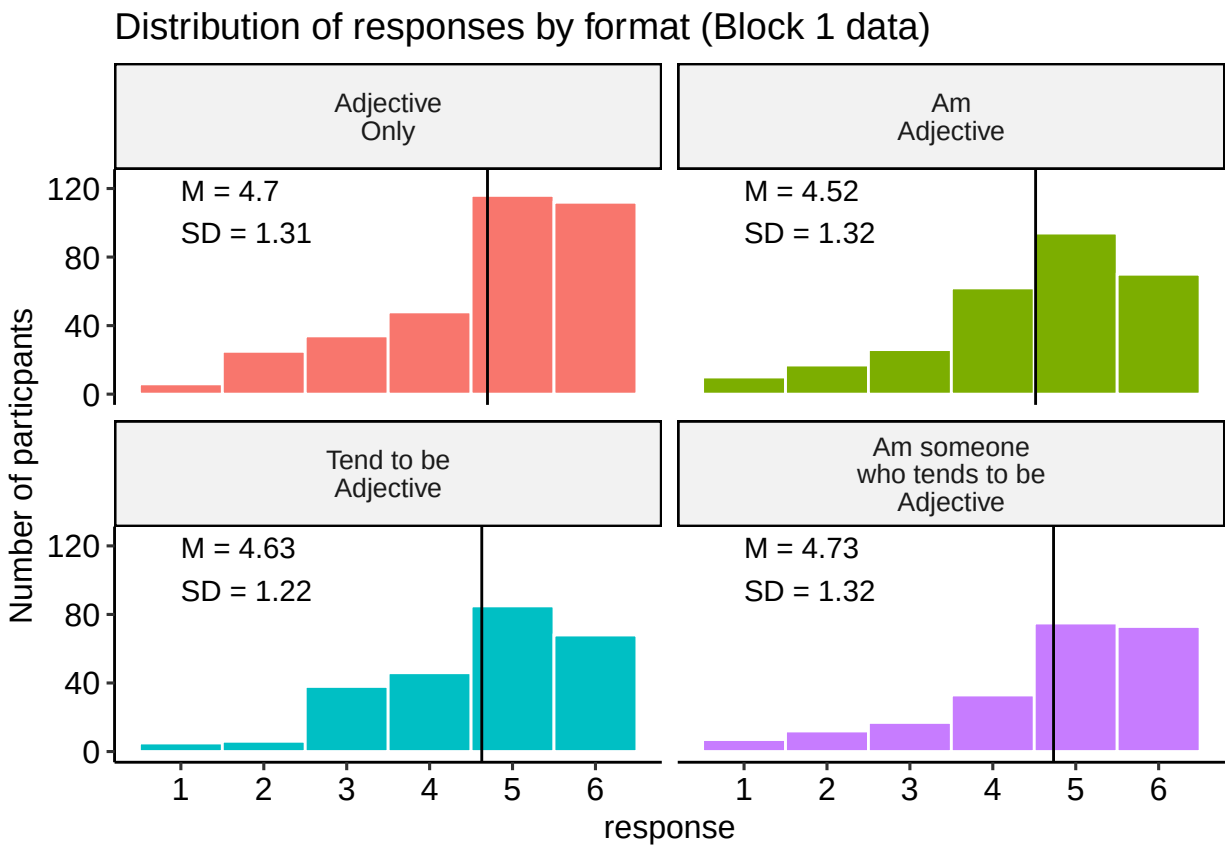


Figure 11: Distribution of responses by category, block 1 data only

Item	Reverse Scored?	SS	MS	df	F	raw	adj
active	N	2.80	0.93	3	0.61	.611	> .999
adventurous	N	1.34	0.45	3	0.50	.682	> .999
broadminded	N	2.27	0.76	3	0.66	.581	> .999
calm	N	3.77	1.26	3	1.29	.295	> .999
caring	N	2.47	0.82	3	1.17	.337	> .999
cautious	N	1.55	0.52	3	0.32	.814	> .999
creative	N	2.35	0.78	3	0.79	.507	> .999
curious	N	0.52	0.17	3	0.15	.927	> .999
friendly	N	0.52	0.17	3	0.34	.796	> .999
hardworking	N	0.03	0.01	3	0.02	.997	> .999
helpful	N	0.20	0.07	3	0.08	.968	> .999
imaginative	N	1.40	0.47	3	0.58	.630	> .999
intelligent	N	1.86	0.62	3	0.44	.725	> .999
lively	N	1.15	0.38	3	0.36	.782	> .999
organized	N	2.20	0.73	3	0.66	.583	> .999
outgoing	N	1.58	0.53	3	0.40	.755	> .999
responsible	N	0.87	0.29	3	0.65	.588	> .999
selfdisciplined	N	2.87	0.96	3	0.72	.545	> .999
softhearted	N	1.25	0.42	3	0.30	.828	> .999
sophisticated	N	1.08	0.36	3	0.21	.887	> .999
sympathetic	N	3.42	1.14	3	1.10	.363	> .999
talkative	N	18.53	6.18	3	3.19	.037	> .999
thorough	N	0.33	0.11	3	0.08	.968	> .999
thrifty	N	8.09	2.70	3	2.06	.126	> .999
warm	N	0.71	0.24	3	0.20	.897	> .999
careless	Y	18.23	6.08	3	2.77	.058	> .999
impulsive	Y	2.65	0.88	3	0.45	.716	> .999
moody	Y	6.21	2.07	3	1.14	.350	> .999
nervous	Y	1.54	0.51	3	0.20	.893	> .999
reckless	Y	5.14	1.71	3	0.65	.587	> .999
worrying	Y	8.95	2.98	3	1.25	.307	> .999

\end{table}

3.1.2 Pairwise t-tests for significant ANOVAs

When format was a significant predictor of response for an item (using the un-adjusted p-value here), we follow up with pairwise comparisons of format. Here we identify the items which meet this criteria. In the manuscript proper, we will only report the results for items in which format was significant, even after applying the Holm correction.

```
sig_item_b1 = summary_by_item_b1 %>%
  filter(p.value < .05)

sig_item_b1 = sig_item_b1$item
sig_item_b1
```

```
## [1] "talkative"
```

Table 6: Differences in response to Talkative by format (Block 1 data only)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-1.49	0.63	31	-2.39	.139
Adjective Only - Tend to be Adjective	-1.40	0.65	31	-2.16	.192
Adjective Only - Am someone who tends to be Adjective	0.01	0.67	31	0.02	> .999
Am Adjective - Tend to be Adjective	0.10	0.68	31	0.14	> .999
Am Adjective - Am someone who tends to be Adjective	1.51	0.70	31	2.15	.192
Tend to be Adjective - Am someone who tends to be Adjective	1.41	0.72	31	1.96	.192

Then we create models for each adjective. We use the `emmeans` package to perform pairwise comparisons, again with a Holm correction on the p-values. We also plot the means and 95% confidence intervals of each mean.

This code will have to be changed after final data collection. It is not self-adapting!

3.1.3 Talkative

The pairwise comparisons of responses to different forms of "talkative" are displayed in Table 6 and Figure 12.

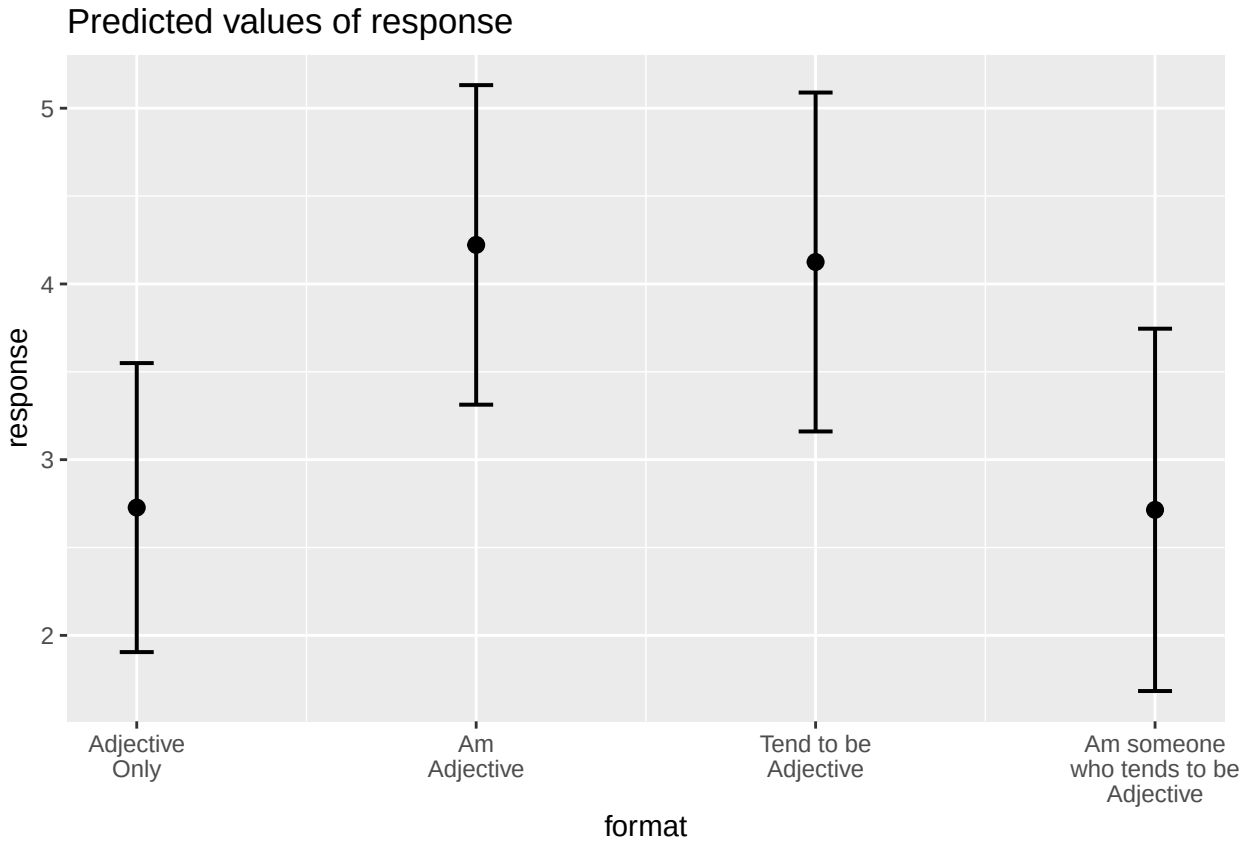


Figure 12: Average response to "talkative" by format (block 1 data only)

3.2 Effect of format (Block 1 and 2)

We again test whether format is a significant predictor of response. However, here we use data from both Blocks 1 and 2. As a reminder, all participants were presented with all four formats during Block 2. We expect this model to have greater power than the previous model, due to both increased sample size (twice as many data points) and because participants now provide data to all four formats, instead of only one (i.e., a within-person analysis).

```
items_12 = items_df %>% filter(block %in% c("1","2"))
```

```
mod.format_b2 = lmer(response~format + (1|proid),
                     data = items_12)
anova(mod.format_b2)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF  DenDF F value Pr(>F)
## format 4.1445  1.3815     3 2104.9  0.9193 0.4307
```

When examining both Block 1 and Block 2 data, item format was unassociated with participants' responses to personality items ($F(3, 2, 104.92) = 0.92, p = .431$). You can see the effect visualized in Figure 13. In addition, the full histogram of responses to each format are presented in Figure 14.

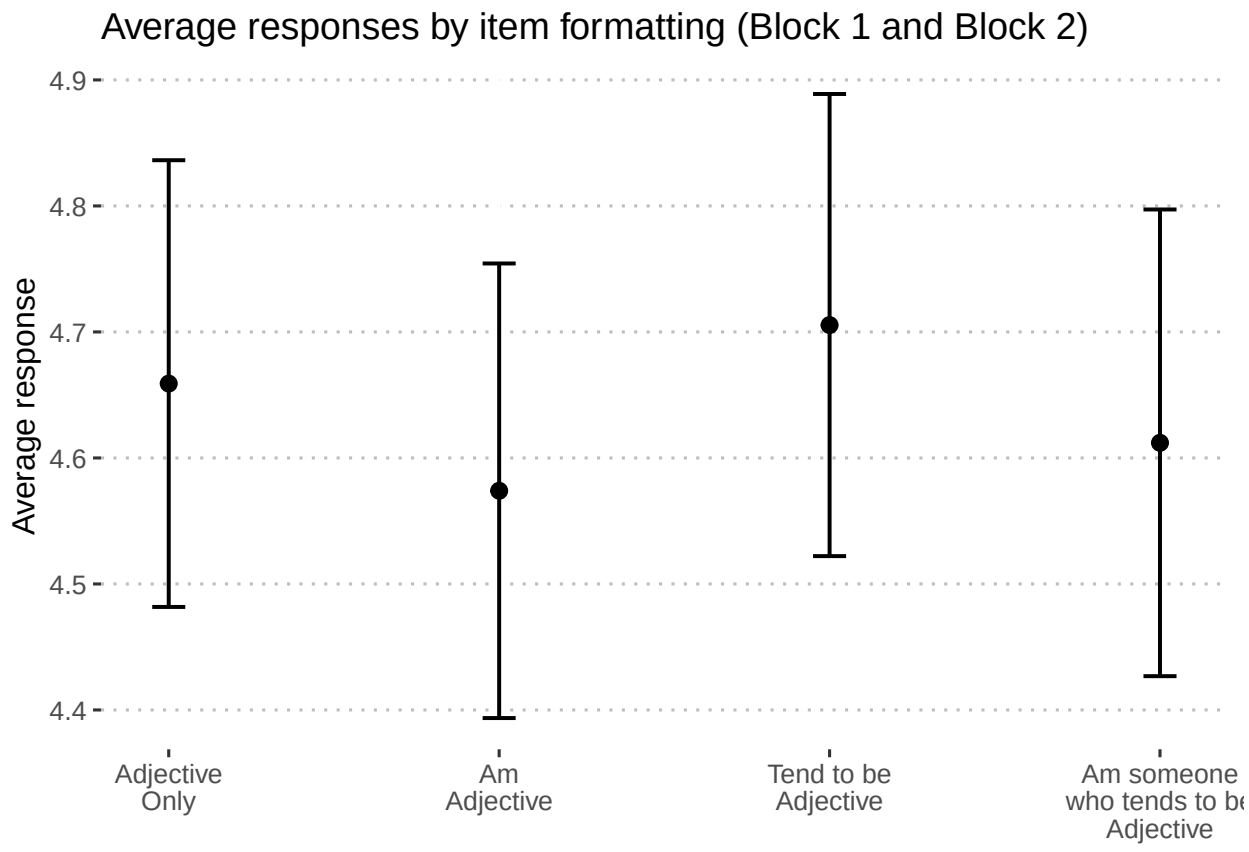


Figure 13: Predicted response on personality items by condition, using Block 1 and Block 2.

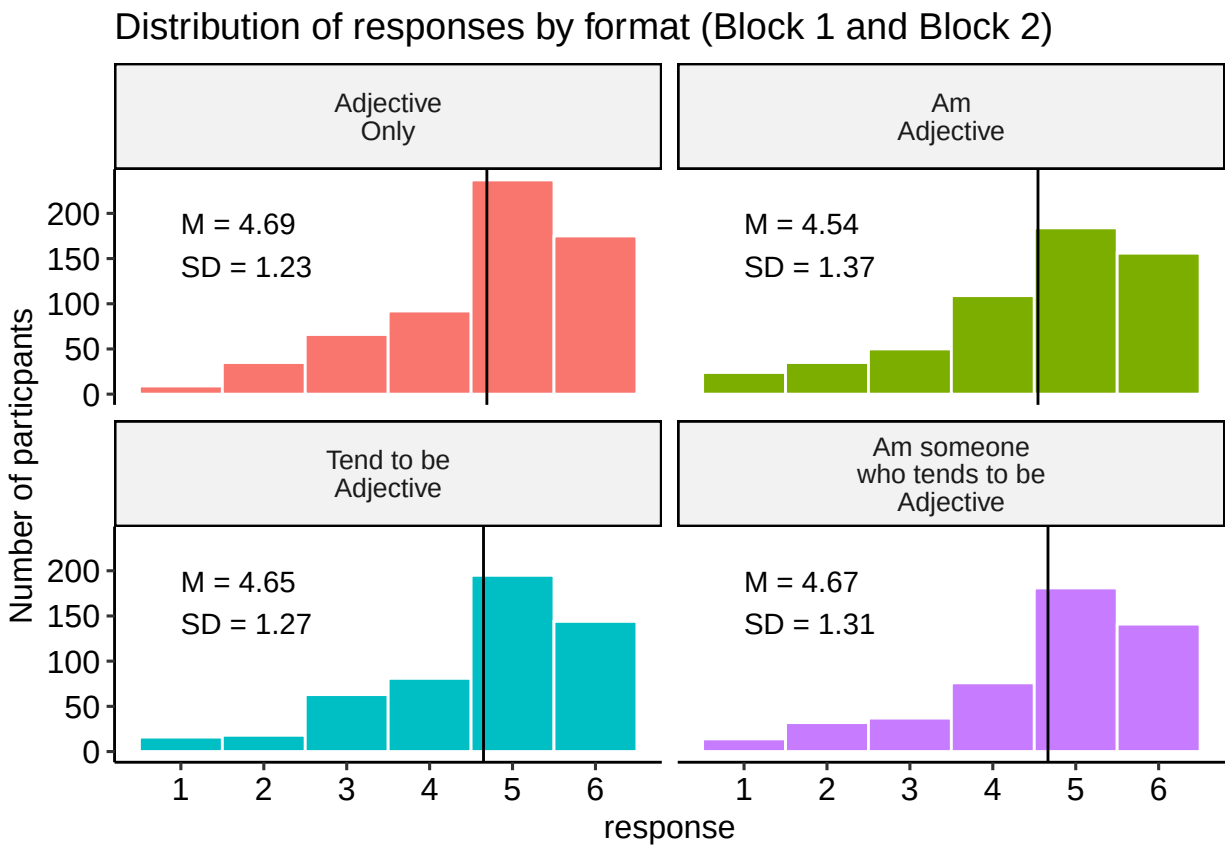


Figure 14: Distribution of responses by category, block 1 and block 2

3.2.1 One model for each adjective

We can also repeat this analysis separately for each trait. We use the `anova` function to estimate the variability due to format and print the corresponding F-test.

```
mod_by_item_b2 = items_12 %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~lmer(response~format + (1|proid), data = .))) %>%
  mutate(aov = map(mod, anova))
```

To present these results, we use the `tidy` function to summarize the findings and extract just the F-test associated with the format variable. We calculate adjusted p-values using a Holm correction. We also create a column that indicates whether the item was reverse-scored; we use this to sort the table, in case a pattern emerges. See the final version of this in Table 7.

3.2.2 Pairwise t-tests for significant ANOVAs

Here we identify the specific items with significant differences.

```
sig_item_b2 = summary_by_item_b2 %>%
  filter(p.value < .05)

sig_item_b2 = sig_item_b2$item
sig_item_b2
```

```
## [1] "careless" "thrifty"
```

Then we create models for each adjective. We use the `emmeans` package to perform pairwise comparisons, again with a Holm correction on the p-values. We also plot the means and 95% confidence intervals of each mean.

This code will have to be changed after final data collection. It is not self-adapting!

3.2.3 Careless

The pairwise comparisons of responses to different forms of "careless" are displayed in Table 8 and Figure 15.

3.2.4 Thrifty

The pairwise comparisons of responses to different forms of "thrifty" are displayed in Table 9 and Figure 16.

3.3 Account for memory effects (Blocks 1 and 2)

One limitation of the two-blocks model is that format effects may depend on a person's memory. For example, suppose that participants, in general, are more likely to respond with a 6 to items containing "tend to" (e.g., "tend to be outgoing") than to items that only start with "am" (e.g., "am outgoing"). However, if a participant remembers that on the first presentation of the item they selected 4, they may be more likely choose 4 again to appear consistent. This example posits that memory moderates format's effect on response. We model this possibility using participant's delayed memory scores, or their recall score 10 minutes after seeing the list of words.

Table 7: Format effects on response by item (block 1 data only)

							p-value	
Item	Reverse Scored?	SS	MS	df1	df2	F	raw	adj
active	N	0.76	0.25	3	40.00	1.01	.398	> .999
adventurous	N	1.51	0.50	3	48.65	1.09	.361	> .999
broadminded	N	1.54	0.51	3	64.49	0.76	.521	> .999
calm	N	0.13	0.04	3	45.02	0.13	.940	> .999
caring	N	1.16	0.39	3	52.66	1.28	.291	> .999
cautious	N	2.30	0.77	3	54.97	0.98	.407	> .999
creative	N	0.14	0.05	3	49.22	0.13	.943	> .999
curious	N	2.35	0.78	3	52.00	1.05	.377	> .999
friendly	N	0.94	0.31	3	46.85	1.45	.239	> .999
hardworking	N	2.10	0.70	3	52.03	2.06	.117	> .999
helpful	N	0.90	0.30	3	60.54	0.82	.488	> .999
imaginative	N	1.03	0.34	3	55.07	0.86	.465	> .999
intelligent	N	1.72	0.57	3	51.16	1.15	.340	> .999
lively	N	1.08	0.36	3	42.77	0.89	.456	> .999
organized	N	0.22	0.07	3	37.39	0.41	.750	> .999
outgoing	N	0.71	0.24	3	39.34	0.83	.484	> .999
responsible	N	1.51	0.50	3	56.50	0.82	.487	> .999
selfdisciplined	N	0.32	0.11	3	41.84	0.26	.853	> .999
softhearted	N	1.95	0.65	3	57.46	0.74	.531	> .999
sophisticated	N	0.05	0.02	3	44.04	0.04	.989	> .999
sympathetic	N	0.49	0.16	3	51.48	0.51	.676	> .999
talkative	N	6.36	2.12	3	43.86	1.95	.135	> .999
thorough	N	2.23	0.74	3	40.16	2.72	.057	> .999
thrifty	N	6.39	2.13	3	51.69	3.30	.027	.849
warm	N	0.10	0.03	3	52.22	0.08	.968	> .999
careless	Y	8.29	2.76	3	53.32	2.84	.047	> .999
impulsive	Y	1.38	0.46	3	41.41	0.85	.473	> .999
moody	Y	1.21	0.40	3	42.77	0.69	.566	> .999
nervous	Y	1.35	0.45	3	46.22	0.45	.716	> .999
reckless	Y	1.14	0.38	3	48.60	0.36	.782	> .999
worrying	Y	1.72	0.57	3	41.67	0.63	.602	> .999

Table 8: Differences in response to Careless by format (Block 1 and Block 2)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	0.82	0.45	58.04	1.83	.359
Adjective Only - Tend to be Adjective	0.14	0.44	54.32	0.32	.749
Adjective Only - Am someone who tends to be Adjective	-0.44	0.44	54.32	-1.01	.633
Am Adjective - Tend to be Adjective	-0.68	0.45	51.43	-1.52	.542
Am Adjective - Am someone who tends to be Adjective	-1.26	0.45	51.43	-2.82	.041
Tend to be Adjective - Am someone who tends to be Adjective	-0.58	0.45	49.48	-1.31	.591

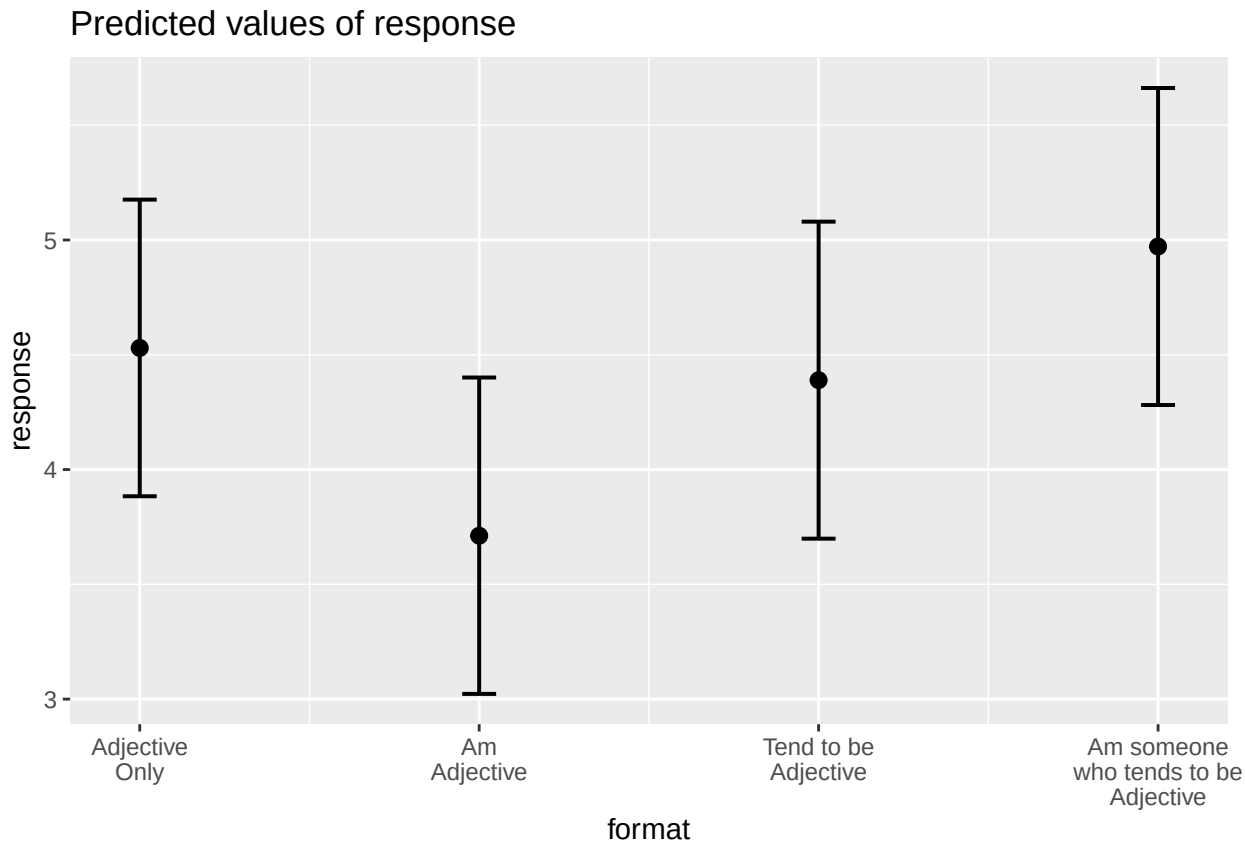


Figure 15: Average response to "careless" by format (Block 1 and Block 2)

Table 9: Differences in response to Thifty by format (Block 1 and Block 2)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-0.02	0.36	56.66	-0.04	> .999
Adjective Only - Tend to be Adjective	-0.92	0.36	55.47	-2.54	.083
Adjective Only - Am someone who tends to be Adjective	-0.09	0.35	52.28	-0.26	> .999
Am Adjective - Tend to be Adjective	-0.90	0.37	56.24	-2.45	.084
Am Adjective - Am someone who tends to be Adjective	-0.08	0.37	54.54	-0.21	> .999
Tend to be Adjective - Am someone who tends to be Adjective	0.83	0.33	43.22	2.49	.084

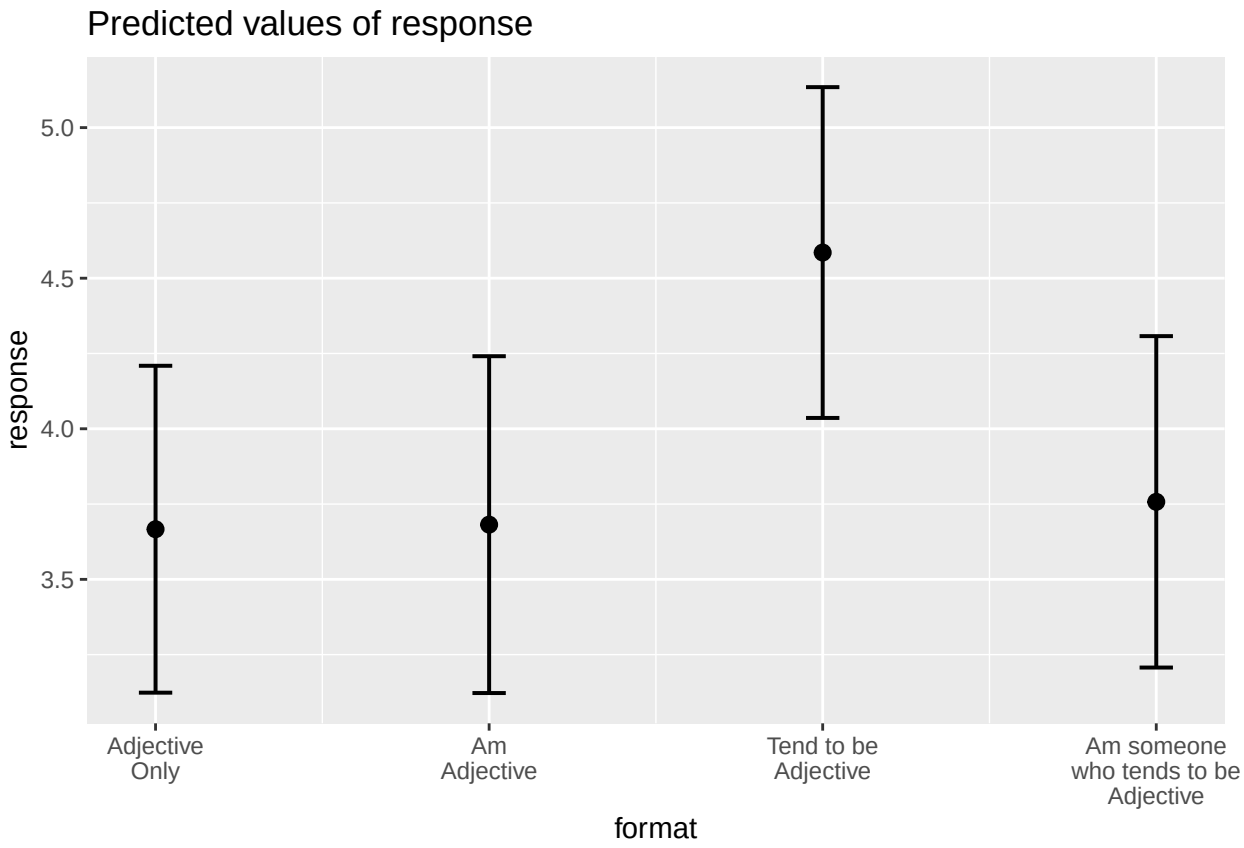


Figure 16: Average response to "thrifty" by format (Block 1 and Block 2)

```
mod.format_mem = lmer(response~format*delayed_memory + (1|proid),
                  data = items_12)
anova(mod.format_mem)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##                        Sum Sq Mean Sq NumDF   DenDF F value Pr(>F)
## format                 4.8144 1.60479     3 2117.22  1.0675 0.3617
## delayed_memory         2.4823 2.48233     1   33.18  1.6513 0.2077
## format:delayed_memory 2.9060 0.96867     3 2118.52  0.6444 0.5865
```

```
summary(mod.format_mem)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: response ~ format * delayed_memory + (1 | proid)
##    Data: items_12
##
## REML criterion at convergence: 7144.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.4343 -0.4565  0.2507  0.6893  1.7916
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  proid    (Intercept) 0.1804   0.4248
##  Residual             1.5033   1.2261
## Number of obs: 2170, groups:  proid, 35
##
## Fixed effects:
##                                                                 Estimate
## (Intercept)                                                      4.58932
## formatAm\nAdjective                                             -0.27329
## formatTend to be\nAdjective                                     -0.07698
## formatAm someone\nwho tends to be\nAdjective                    -0.14891
## delayed_memory                                                   0.01320
## formatAm\nAdjective:delayed_memory                               0.03632
## formatTend to be\nAdjective:delayed_memory                       0.02519
## formatAm someone\nwho tends to be\nAdjective:delayed_memory      0.02048
##                                                                Std. Error
## (Intercept)                                                      0.18198
## formatAm\nAdjective                                              0.15843
## formatTend to be\nAdjective                                      0.16694
## formatAm someone\nwho tends to be\nAdjective                     0.17237
## delayed_memory                                                   0.03108
## formatAm\nAdjective:delayed_memory                               0.02641
## formatTend to be\nAdjective:delayed_memory                       0.02944
## formatAm someone\nwho tends to be\nAdjective:delayed_memory      0.02956
##                                                                      df t value
## (Intercept)                                                       62.17156  25.219
## formatAm\nAdjective                                             2128.91591  -1.725
## formatTend to be\nAdjective                                     2092.17673  -0.461
```

```
## formatAm someone\nwho tends to be\nAdjective                2139.15284  -0.864
## delayed_memory                                               63.52491   0.425
## formatAm\nAdjective:delayed_memory                         2116.48721   1.375
## formatTend to be\nAdjective:delayed_memory                 2070.60879   0.856
## formatAm someone\nwho tends to be\nAdjective:delayed_memory 2147.54209   0.693
##                                                            Pr(>|t|)
## (Intercept)                                                  <2e-16 ***
## formatAm\nAdjective                                          0.0847 .
## formatTend to be\nAdjective                                  0.6447
## formatAm someone\nwho tends to be\nAdjective                 0.3878
## delayed_memory                                               0.6725
## formatAm\nAdjective:delayed_memory                           0.1692
## formatTend to be\nAdjective:delayed_memory                   0.3922
## formatAm someone\nwho tends to be\nAdjective:delayed_memory   0.4885
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) frmtAA frTtbA frAswttbA dlyd_m frAA:_ fTtbA:
## frmtAmAdjct -0.439
## frmtTndtbAd -0.433  0.478
## frmtAswttbA -0.407  0.455  0.470
## delayd_mmry -0.869  0.387  0.381  0.355
## frmtAAdjc:_  0.397 -0.859 -0.433 -0.413    -0.465
## frmtTtbAd:_  0.370 -0.409 -0.865 -0.398    -0.431  0.495
## frAswttbA:_  0.354 -0.400 -0.410 -0.873    -0.410  0.484  0.460
```

When examining both Block 1 and Block 2 data, memory did not have a main effect on participant responses ($F(1, 33.18) = 1.65, p = .208$) and did not moderate differences between formats ($F(3, 2, 118.52) = 0.64, p = .586$). See these results in Figure 17.

```
plot_model(mod.format_mem,
           type = "pred",
           term = c("format", "delayed_memory[meansd]")) +
  geom_line() +
  labs(x = NULL,
       y = "Average response") +
  scale_color_discrete("Memory", labels = c("-1SD", "Mean", "+1SD"))+
  theme_pubclean()
```

3.3.1 One model for each adjective

Again, we test this model within each trait adjective, to determine whether the moderating effect of memory is stronger for any particular trait(s). The results are summarized in Table **??**.

3.3.2 Pairwise t-tests for significant ANOVAs

Here we identify the specific items with significant differences.

```
sig_item_mem = summary_by_item_mem %>%
  filter(p.value < .05)
```

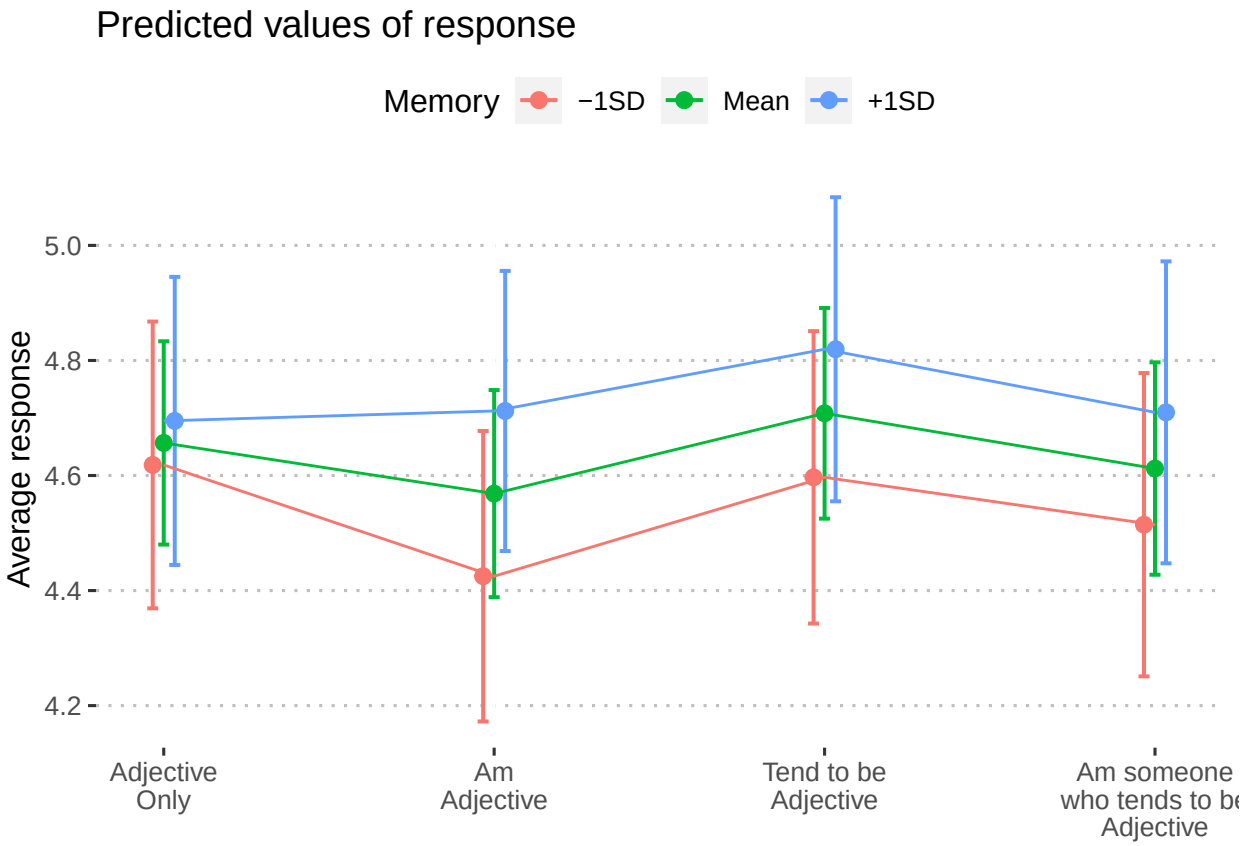


Figure 17: Predicted response on personality items by condition after controlling for delayed_memory.

						p-value	
Item	Reverse Scored?	SS	MS	df	F	raw	adj
active	N	0.29	0.10	3	0.06	.980	> .999
adventurous	N	1.17	0.39	3	0.39	.759	> .999
broadminded	N	0.71	0.24	3	0.28	.843	> .999
calm	N	0.15	0.05	3	0.06	.980	> .999
caring	N	0.16	0.05	3	0.08	.968	> .999
cautious	N	0.42	0.14	3	0.12	.948	> .999
creative	N	1.12	0.37	3	0.44	.727	> .999
curious	N	1.28	0.43	3	0.27	.843	> .999
friendly	N	1.83	0.61	3	1.01	.394	> .999
hardworking	N	0.74	0.25	3	0.37	.771	> .999
helpful	N	1.31	0.44	3	0.73	.536	> .999
imaginative	N	0.65	0.22	3	0.35	.790	> .999
intelligent	N	2.94	0.98	3	0.95	.421	> .999
lively	N	1.21	0.40	3	0.28	.840	> .999
organized	N	5.06	1.69	3	1.53	.216	> .999
outgoing	N	1.72	0.57	3	0.40	.753	> .999
responsible	N	2.53	0.84	3	1.11	.353	> .999
selfdisciplined	N	2.29	0.76	3	0.60	.616	> .999
softhearted	N	0.46	0.15	3	0.13	.943	> .999
sophisticated	N	4.21	1.40	3	0.89	.451	> .999
sympathetic	N	1.11	0.37	3	0.40	.754	> .999
talkative	N	8.53	2.84	3	1.23	.306	> .999
thorough	N	3.87	1.29	3	1.13	.344	> .999
thrifty	N	0.61	0.20	3	0.13	.940	> .999
warm	N	1.65	0.55	3	0.69	.560	> .999
careless	Y	3.74	1.25	3	0.55	.653	> .999
impulsive	Y	7.37	2.46	3	1.13	.343	> .999
moody	Y	10.15	3.38	3	1.76	.165	> .999
nervous	Y	4.40	1.47	3	0.57	.638	> .999
reckless	Y	7.65	2.55	3	1.01	.394	> .999
worrying	Y	1.77	0.59	3	0.22	.880	> .999

```
sig_item_mem = sig_item_mem$item
sig_item_mem
```

```
## character(0)
```

3.4 Inclusion of "I" (Block 1 and Block 3)

Finally, we test whether the inclusion of the word "I" impacts item response (e.g. "I am outgoing"). We used two multilevel models, nesting response within participant to account for dependence. Our primary predictors are format and also the presence of the word "I". Because we have no specific rationale for how or why "I" would impact responses, we test both the partialled main effect of "I" as well as the interaction with format. Here, we use data from Blocks 1 and 3. Results are presented in Figure 18 and the full distribution of responses by format and "i" are presented in Figure 19.

```
items_13 = items_df %>%
  filter(block %in% c("1","3")) %>%
  filter(condition != "A") %>%
  filter(time2 == "yes")
```

```
mod.format_b3_1 = lmer(response~format + i + (1|proid),
                  data = items_13)
anova(mod.format_b3_1)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##        Sum Sq Mean Sq NumDF  DenDF F value Pr(>F)
## format 2.4558  1.2279     2  13.00  0.8151 0.4640
## i      3.4314  3.4314     1 979.33  2.2778 0.1316
```

```
mod.format_b3_2 = lmer(response~format*i + (1|proid),
                  data = items_13)
anova(mod.format_b3_2)
```

```
## Type III Analysis of Variance Table with Satterthwaite's method
##          Sum Sq Mean Sq NumDF  DenDF F value Pr(>F)
## format   3.3194 1.65969     2  14.78  1.1006 0.3584
## i        1.7404 1.74044     1 976.53  1.1542 0.2829
## format:i 1.5669 0.78345     2 976.81  0.5195 0.5950
```

```
means_by_group = items_13 %>%
  group_by(format, i) %>%
  summarise(m = mean(response),
            s = sd(response))

items_13 %>%
  ggplot(aes(x = response, fill = i)) +
  geom_histogram(bins = 6, color = "white") +
  geom_vline(aes(xintercept = m), data = means_by_group) +
  geom_text(aes(x = 1,
                y = 100,
```

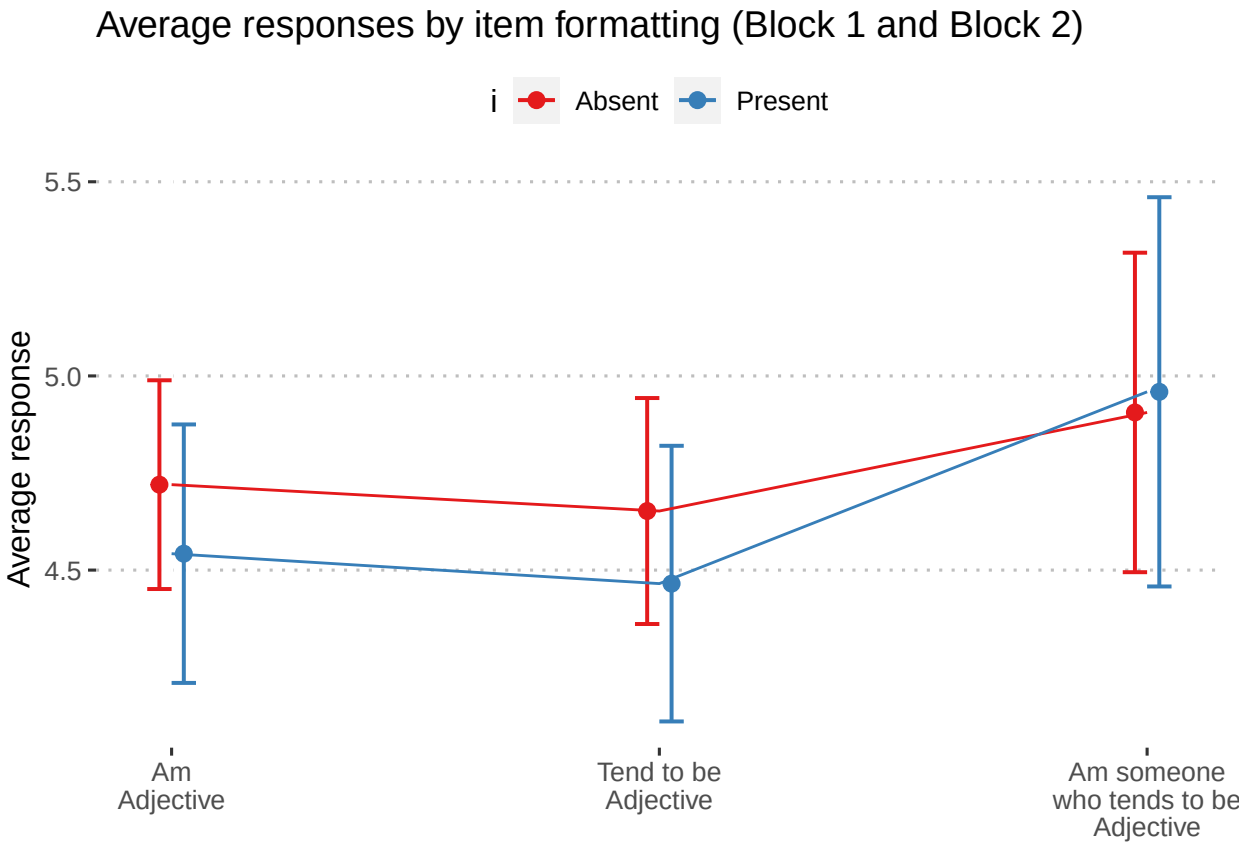


Figure 18: Predicted response on personality items by condition, using only Block 1 data.

```
              label = paste("M =", round(m,2),
                            "\nSD =", round(s,2))),
            data = means_by_group,
            hjust =0,
            vjust = 1) +
  facet_grid(i~format, scales = "free") +
  guides(fill = "none") +
  scale_x_continuous(breaks = 1:6) +
  labs(y = "Number of particpants",
       title = "Distribution of responses by format (Block 1 and Block 2)") +
  theme_pubr()
```

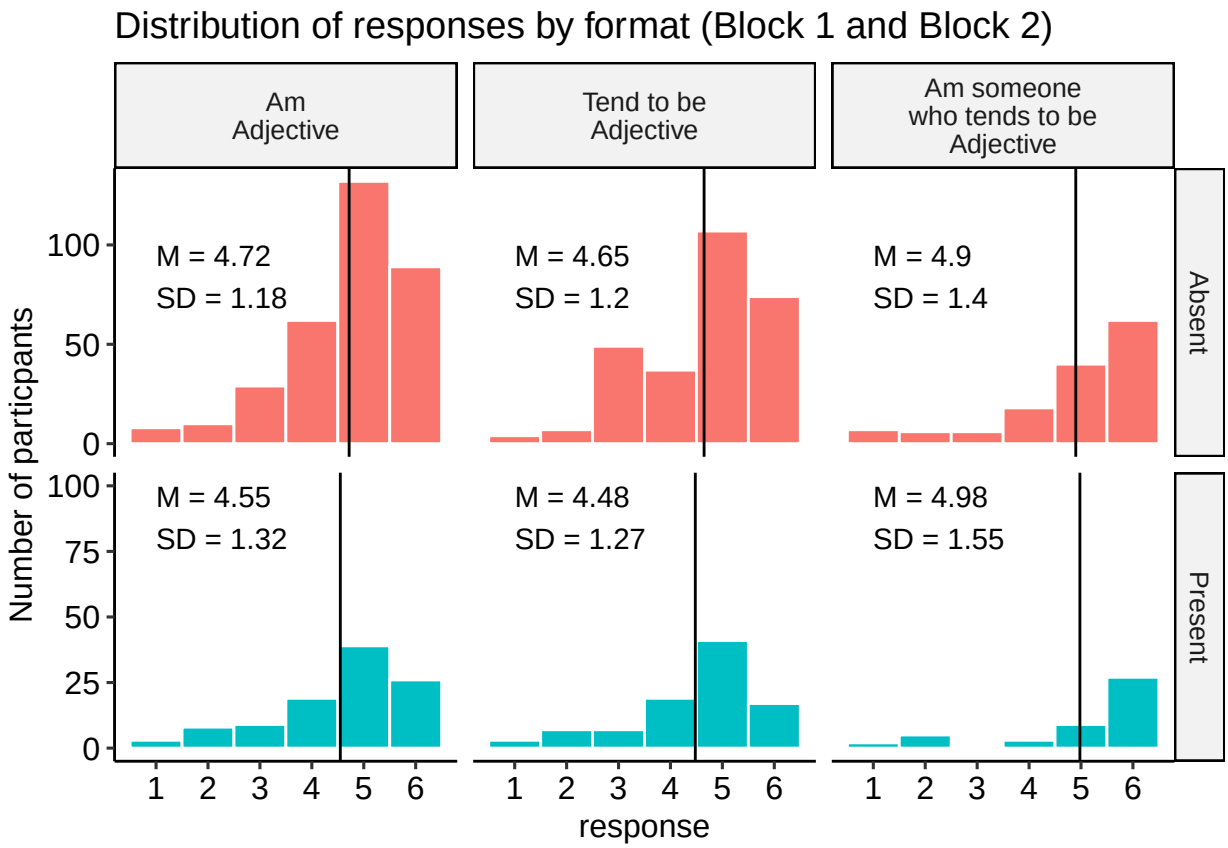


Figure 19: Distribution of responses by category, block 1 and block 2

3.4.1 One model for each adjective

As before, we test both the additive (Table 10) and interaction (Table 11) terms of `format` and `i` for each item.

```
mod_by_item_i = items_13 %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~lmer(response~format + i + (1|proid), data = .))) %>%
  mutate(aov = map(mod, anova))
```

Table 10: Effect of "I" (block 1 and 3 data)

							p-value	
Item	Reverse Scored?	SS	MS	df1	df2	F	raw	adj
active	N	0.79	0.79	1	17.53	2.15	.161	> .999
adventurous	N	0.69	0.69	1	28.00	1.31	.262	> .999
broadminded	N	0.07	0.07	1	21.24	0.13	.718	> .999
calm	N	2.26	2.26	1	21.01	2.85	.106	> .999
caring	N	0.19	0.19	1	28.00	0.63	.433	> .999
cautious	N	0.59	0.59	1	18.73	0.97	.337	> .999
creative	N	0.53	0.53	1	16.63	2.22	.155	> .999
curious	N	0.16	0.16	1	19.78	0.20	.664	> .999
friendly	N	0.00	0.00	1	28.00	0.00	.971	> .999
hardworking	N	0.00	0.00	1	17.35	0.04	.850	> .999
helpful	N	0.11	0.11	1	23.35	0.70	.410	> .999
imaginative	N	0.08	0.08	1	20.83	0.31	.582	> .999
intelligent	N	0.02	0.02	1	18.94	0.10	.755	> .999
lively	N	4.35	4.35	1	18.29	13.11	.002	.059
organized	N	0.29	0.29	1	16.69	0.91	.354	> .999
outgoing	N	0.18	0.18	1	17.41	0.62	.440	> .999
responsible	N	0.24	0.24	1	21.98	0.74	.399	> .999
selfdisciplined	N	0.17	0.17	1	17.30	1.13	.303	> .999
softhearted	N	0.91	0.91	1	21.21	0.69	.415	> .999
sophisticated	N	0.45	0.45	1	17.51	0.79	.385	> .999
sympathetic	N	0.60	0.60	1	20.31	1.17	.293	> .999
talkative	N	0.42	0.42	1	16.66	0.77	.394	> .999
thorough	N	1.64	1.64	1	20.01	5.48	.030	.893
thrifty	N	1.48	1.48	1	19.77	1.78	.197	> .999
warm	N	0.00	0.00	1	20.23	0.00	.967	> .999
careless	Y	1.34	1.34	1	26.28	0.60	.446	> .999
impulsive	Y	0.13	0.13	1	17.64	0.16	.693	> .999
moody	Y	0.39	0.39	1	20.49	0.40	.534	> .999
nervous	Y	3.24	3.24	1	26.54	1.89	.181	> .999
reckless	Y	0.85	0.85	1	21.68	0.68	.420	> .999
worrying	Y	0.01	0.01	1	20.29	0.01	.932	> .999

```
mod_by_item_i2 = items_13 %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~lmer(response~format*i + (1|proid), data = .))) %>%
  mutate(aov = map(mod, anova))
```

3.4.2 Pairwise t-tests for significant ANOVAs

Here we identify the specific items with significant differences.

```
sig_item_i = summary_by_item_i %>%
  filter(p.value < .05)

sig_item_i = sig_item_i$item
sig_item_i
```

```
## [1] "lively"   "thorough"
```

3.4.3 Curious

The pairwise comparisons of responses to different forms of "curious" are displayed in Table 12 and Figure ??.

```
curious_model_i = items_13 %>%
  filter(item == "curious") %>%
  lmer(response~format*i + (1|proid),
                              data = .)
```

Table 11: Interaction of format and "I" (block 1 and 3 data)

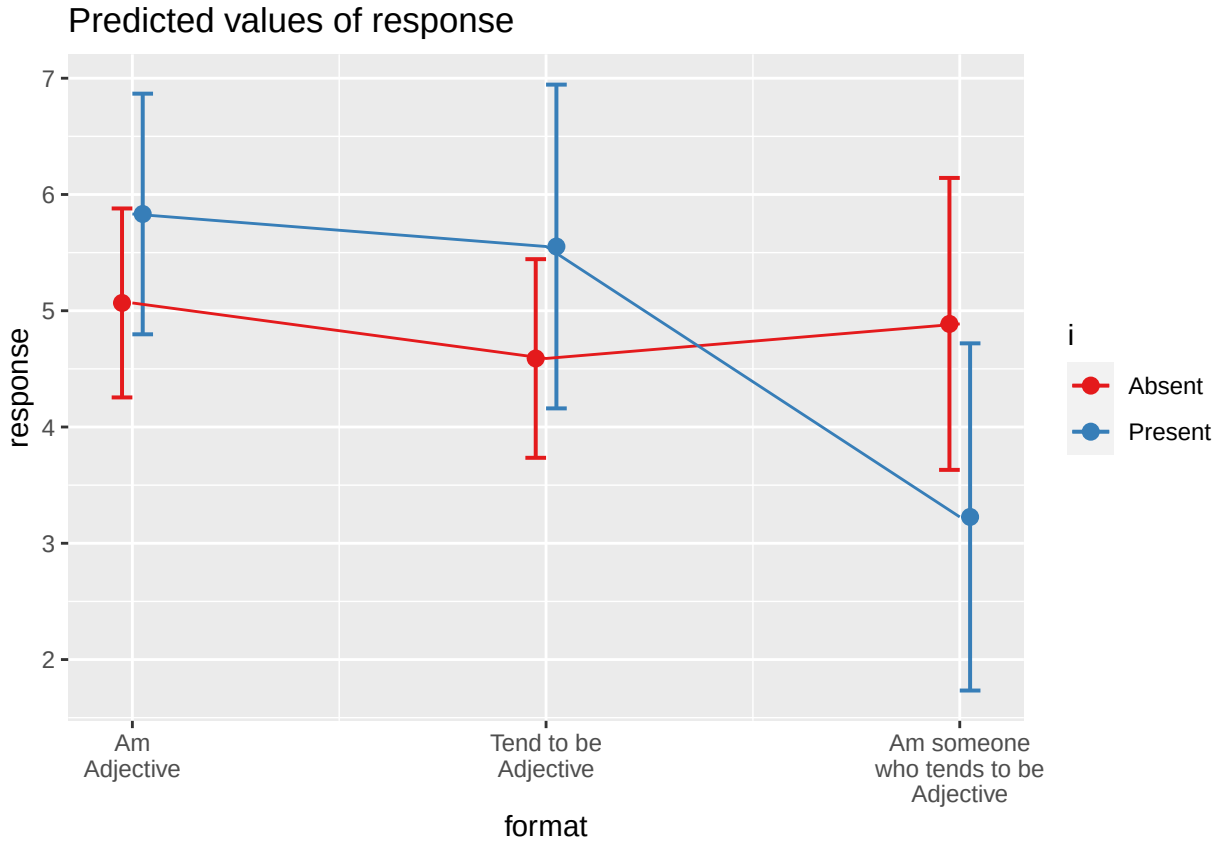
							p-value	
Item	Reverse Scored?	SS	MS	df1	df2	F	raw	adj
active	N	0.65	0.32	2	15.11	0.76	.487	> .999
adventurous	N	1.49	0.75	2	22.11	1.52	.241	> .999
broadminded	N	1.95	0.97	2	20.88	1.94	.169	> .999
calm	N	0.31	0.16	2	20.67	0.18	.837	> .999
caring	N	0.27	0.13	2	26.00	0.43	.658	> .999
cautious	N	1.23	0.62	2	15.17	1.15	.344	> .999
creative	N	0.11	0.05	2	15.03	0.21	.812	> .999
curious	N	5.15	2.58	2	16.43	4.65	.025	.729
friendly	N	0.55	0.27	2	26.00	0.87	.431	> .999
hardworking	N	0.28	0.14	2	14.65	1.16	.342	> .999
helpful	N	0.23	0.12	2	21.45	0.69	.515	> .999
imaginative	N	0.29	0.15	2	19.61	0.51	.608	> .999
intelligent	N	0.07	0.04	2	16.71	0.16	.854	> .999
lively	N	2.34	1.17	2	15.12	5.78	.014	.413
organized	N	0.33	0.16	2	15.11	0.48	.629	> .999
outgoing	N	0.26	0.26	1	16.27	0.90	.356	> .999
responsible	N	2.07	1.03	2	19.58	4.17	.031	.869
selfdisciplined	N	0.47	0.23	2	14.86	1.65	.225	> .999
softhearted	N	10.15	5.08	2	20.14	3.69	.043	> .999
sophisticated	N	1.96	0.98	2	15.26	1.96	.174	> .999
sympathetic	N	3.24	1.62	2	14.45	5.91	.013	.413
talkative	N	0.17	0.08	2	14.68	0.13	.875	> .999
thorough	N	0.01	0.00	2	18.49	0.01	.988	> .999
thrifty	N	1.22	0.61	2	17.82	0.70	.509	> .999
warm	N	0.45	0.23	2	18.90	0.40	.673	> .999
careless	Y	9.07	4.54	2	24.69	2.16	.137	> .999
impulsive	Y	0.06	0.03	2	15.73	0.03	.966	> .999
moody	Y	1.12	0.56	2	17.91	0.55	.587	> .999
nervous	Y	1.35	0.67	2	23.48	0.41	.667	> .999
reckless	Y	2.37	1.18	2	18.59	1.05	.370	> .999
worrying	Y	1.95	0.97	2	18.56	0.89	.429	> .999

Table 12: Interaction of format and "I"

Term	Estimate	SE	t	df	p
(Intercept)	5.07	0.41	12.22	15.50	< .001
Tend to be Adjective	-0.48	0.60	-0.79	14.75	.440
Am someone who tends to be Adjective	-0.18	0.76	-0.24	15.96	.816
I	0.77	0.50	1.53	16.12	.145
Tend to be Adjective:I	0.20	0.85	0.23	17.12	.819
Am someone who tends to be Adjective:I	-2.43	0.87	-2.78	15.68	.013

Table 13: Interaction of format and "I"

Term	Estimate	SE	t	df	p
(Intercept)	4.87	0.36	13.58	14.04	< .001
Tend to be Adjective	0.22	0.53	0.42	14.15	.681
Am someone who tends to be Adjective	0.65	0.65	1.00	13.85	.334
I	-1.38	0.36	-3.86	15.06	.002
Tend to be Adjective:I	0.02	0.50	0.05	14.95	.963
Am someone who tends to be Adjective:I	2.27	0.71	3.20	15.27	.006



3.4.4 Lively

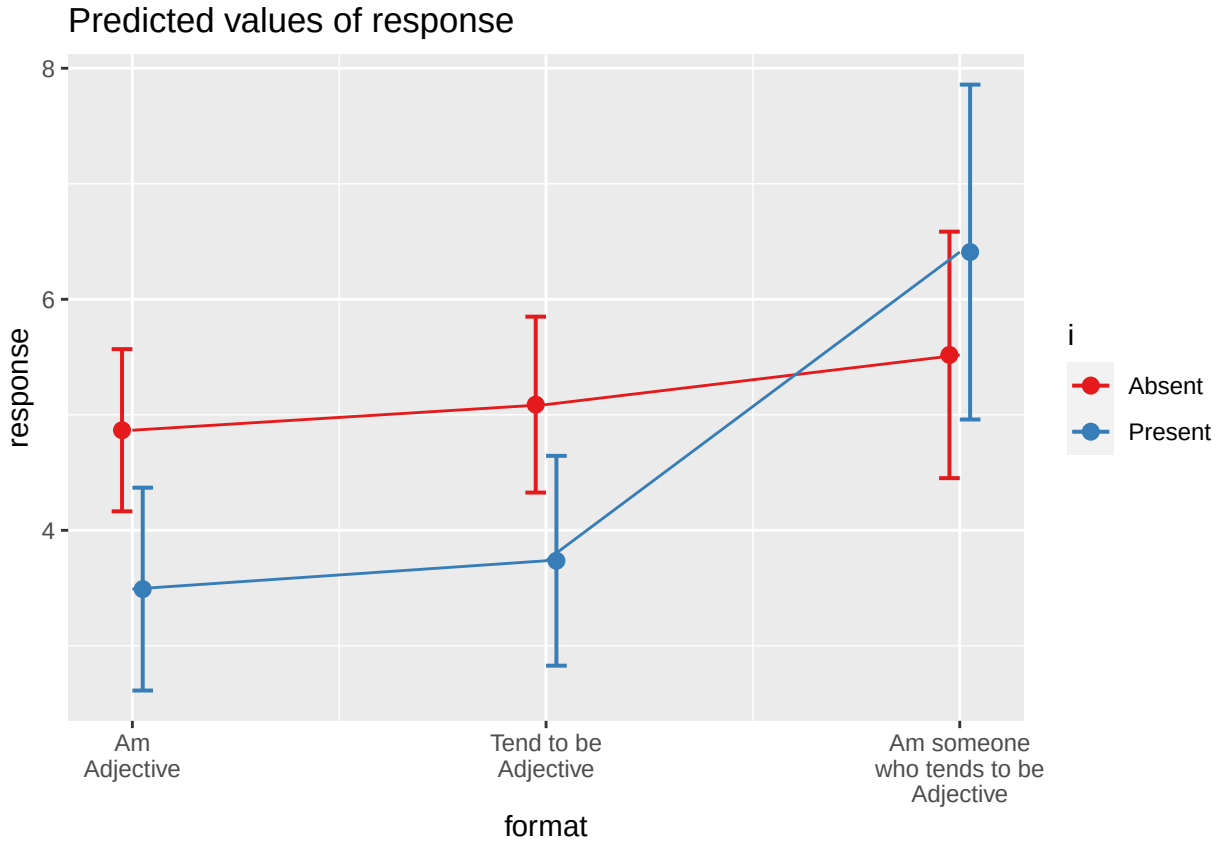
The pairwise comparisons of responses to different forms of "lively" are displayed in Table 13 and Figure **??**.

```
lively_model_i = items_13 %>%
  filter(item == "lively") %>%
  lmer(response~format*i + (1|proid),
                              data = .)
```

```
plot_model(lively_model_i, type = "pred",
           terms = c("format", "i")) +
  geom_line()
```

Table 14: Interaction of format and "I"

Term	Estimate	SE	t	df	p
(Intercept)	5.50	0.21	26.30	16.71	< .001
Tend to be Adjective	-0.04	0.31	-0.11	17.32	.910
Am someone who tends to be Adjective	0.06	0.37	0.17	15.13	.868
I	0.25	0.32	0.78	18.85	.443
Tend to be Adjective:I	-1.14	0.46	-2.50	18.26	.022
Am someone who tends to be Adjective:I	0.38	0.69	0.55	20.83	.591



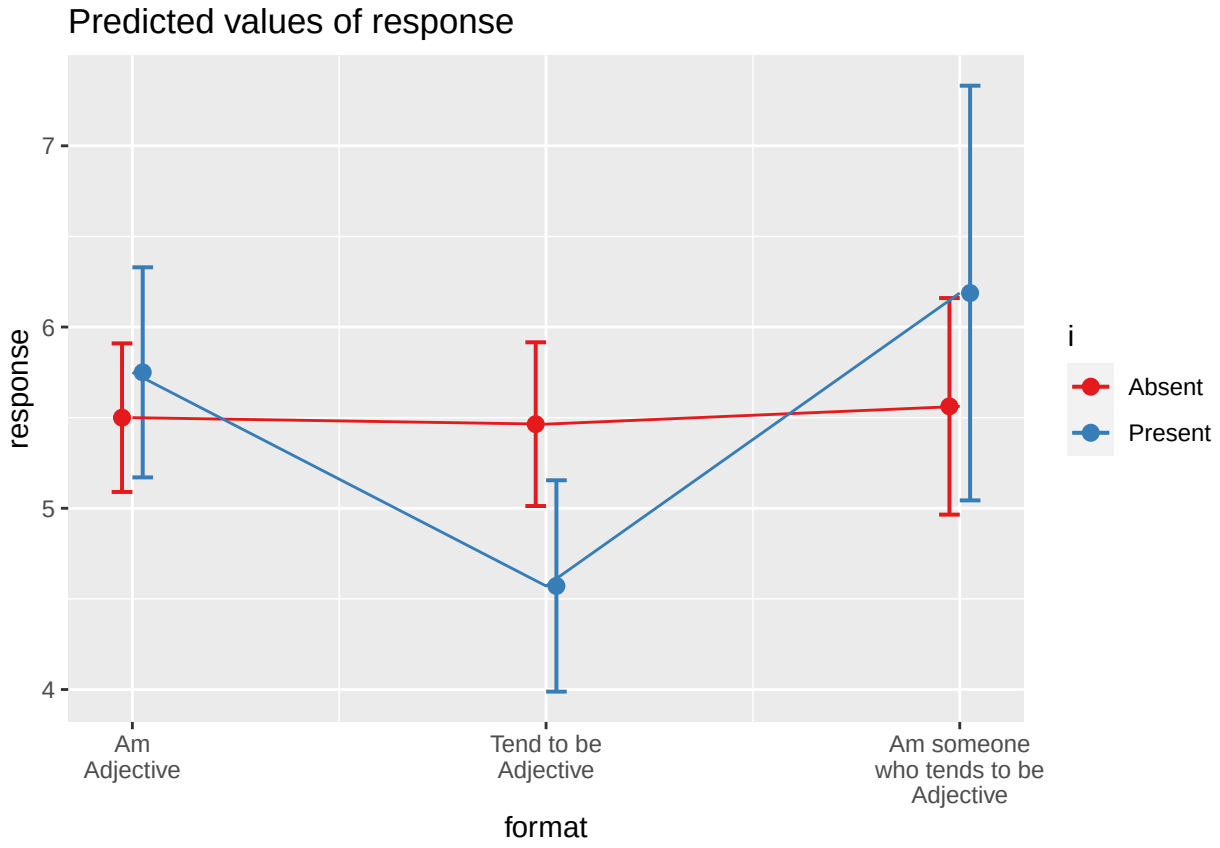
3.4.5 Responsible

The pairwise comparisons of responses to different forms of "responsible" are displayed in Table 14 and Figure **??**.

```
responsible_model_i = items_13 %>%
  filter(item == "responsible") %>%
  lmer(response~format*i + (1|proid),
                              data = .)
```

Table 15: Interaction of format and "I"

Term	Estimate	SE	t	df	p
(Intercept)	4.73	0.37	12.67	14.07	< .001
Tend to be Adjective	0.27	0.54	0.50	13.60	.624
Am someone who tends to be Adjective	-0.50	0.68	-0.73	14.34	.480
I	-0.30	0.36	-0.82	14.25	.423
Tend to be Adjective:I	0.30	0.61	0.48	14.87	.637
Am someone who tends to be Adjective:I	2.10	0.62	3.37	13.99	.005

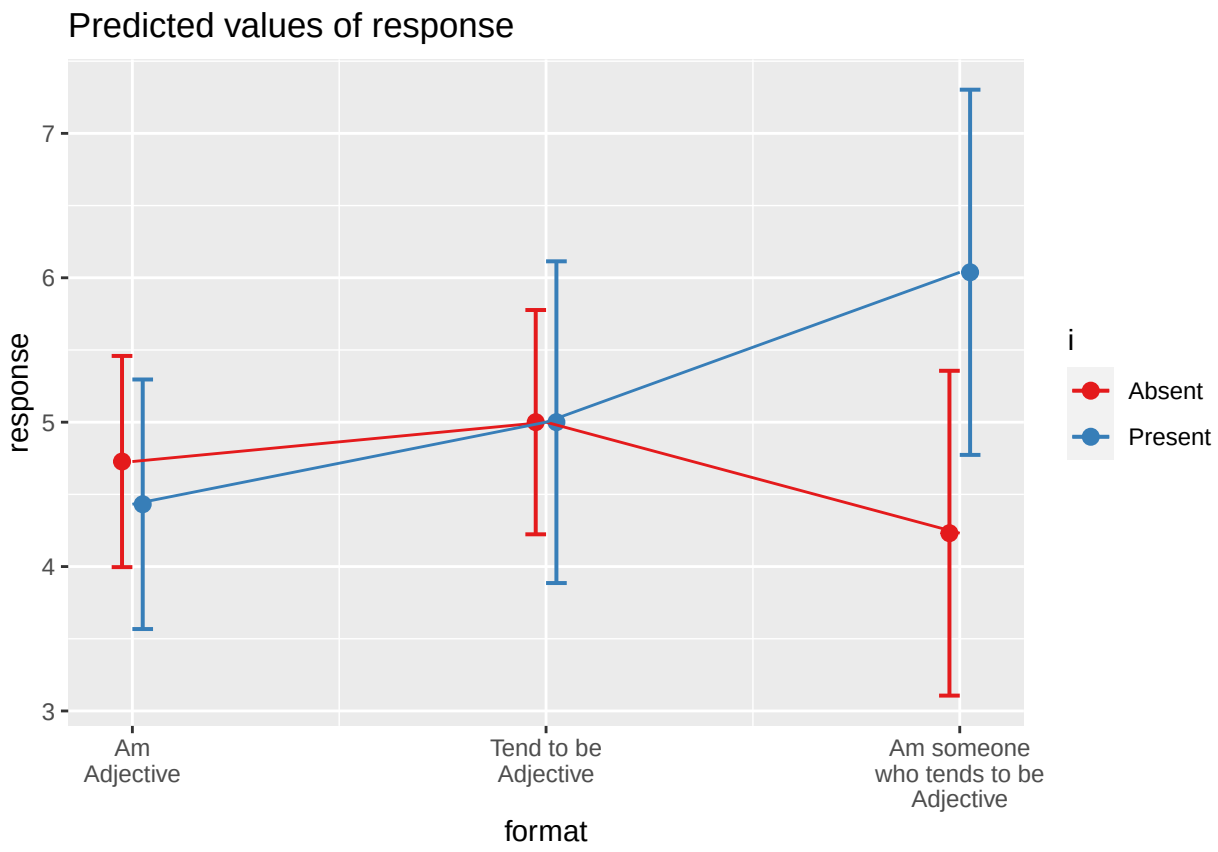


3.4.6 Sympathetic

The pairwise comparisons of responses to different forms of "sympathetic" are displayed in Table 15 and Figure ??.

```
sympathetic_model_i = items_13 %>%
  filter(item == "sympathetic") %>%
  lmer(response~format*i + (1|proid),
                              data = .)
```

```
plot_model(sympathetic_model_i, type = "pred",
           terms = c("format", "i")) +
  geom_line()
```


Predicted values of response
response
7
6
5
4
3
Am
Adjective
Tend to be
Adjective
Am someone
who tends to be
Adjective
format
i
Absent
Present