Supplemental file

Last updated 2022-07-05

Contents

0.1 Description of file

Analyses – including data cleaning, descriptive statistics, and power estimates – for this project were documented using a series of RMarkdown (.Rmd) files. This document aggregates all files, in the order in which they are meant to be run, into a single RMarkdown file and compiles the output into a single PDF. Those interested in reproducing this document should do the following:

- Check that LaTex has been installed on their machine.
- Create an RStudio project to store the data and scripts included on this OSF page.
- Download the supplementary workspace (scripts and data) as they are organized on the OSF page – specifically this means including data in a folder called "deidentified data" and scripts in a folder called "scripts." These folders should be saved in the RStudio project directory.
- Check that the file called `renv.lock` is downloaded and located in the RStudio project folder. This contains a snapshot of the packages and their versions used in this project.

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(glmmTMB) # for multilevel modeling
library(broom) # for presenting results
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)

  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 19, 2021_18.30.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.

```r
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.

```r
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.

```r
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".

```r
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.

```r
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.

```r
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
```

```r
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 S1 for a visualization of the spread and associations between run lengths across participants.

```r
#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.

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

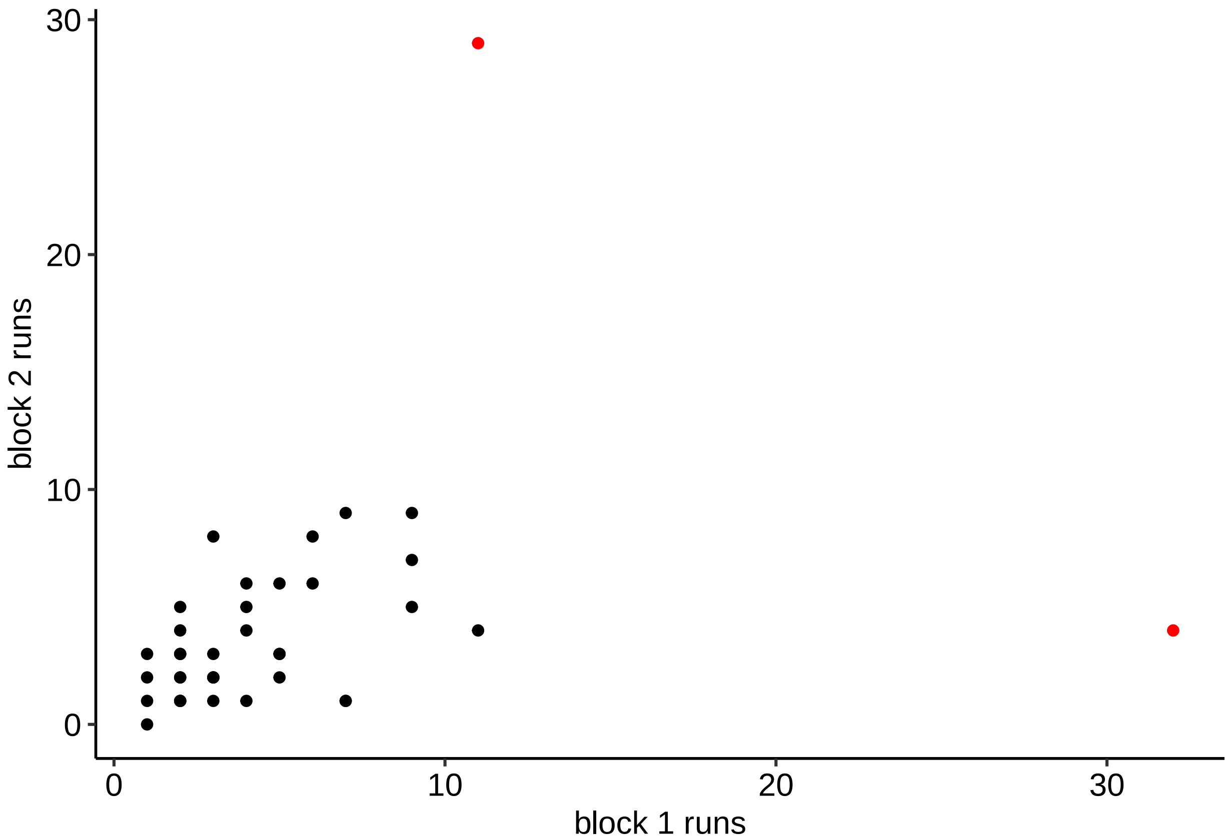


Figure S1: Maximum number of same consecutive responses in personality blocks.

```
  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 S2 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.

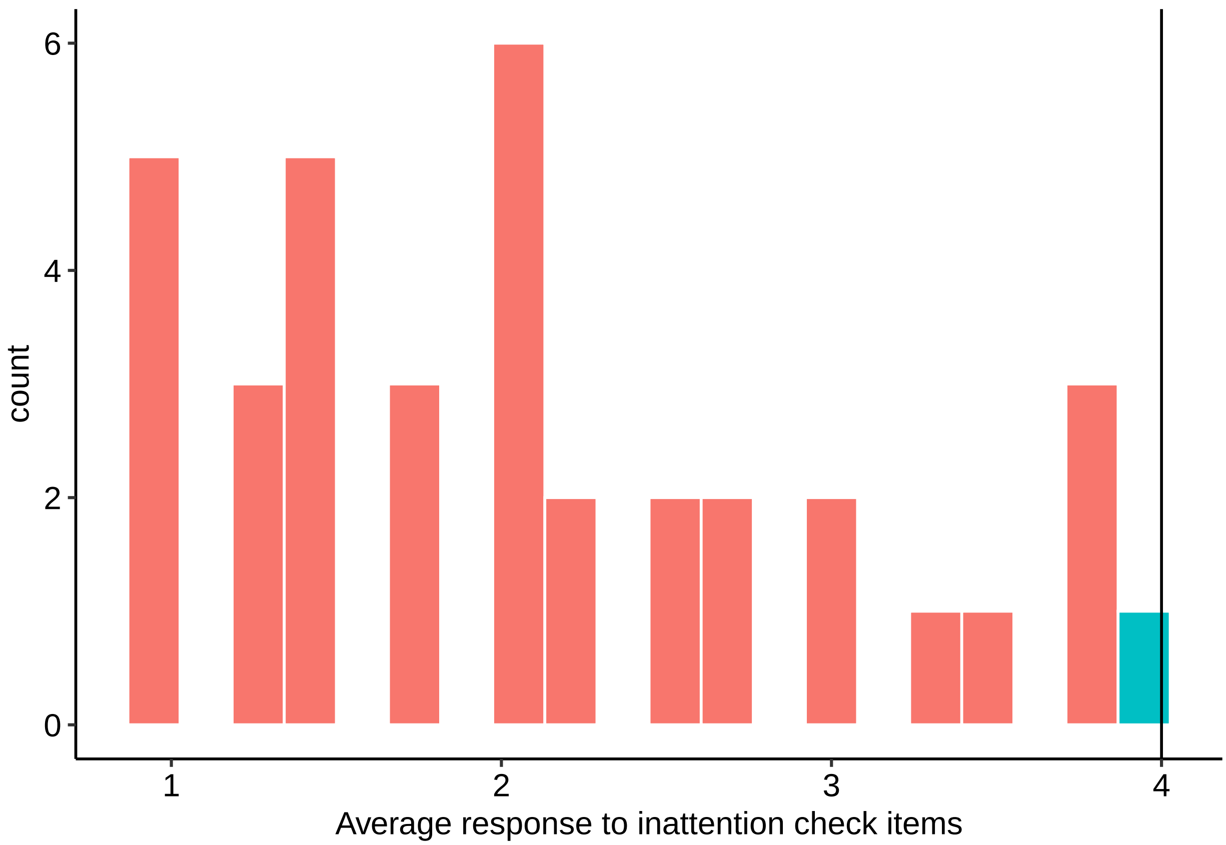


Figure S2: Average response to inattention check items

```r
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 S3 for a distribution of average response time.

```r
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"
  ))
```

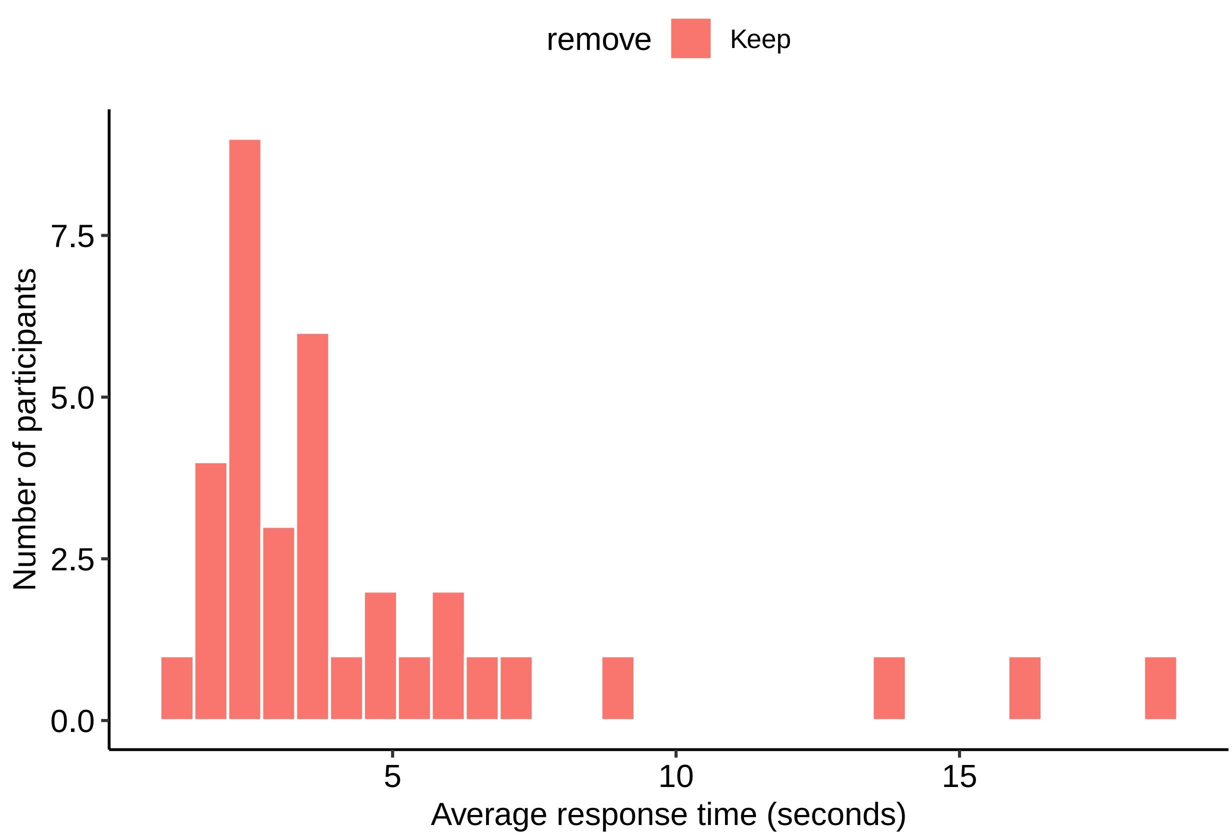


Figure S3: Distribution of average time to respond to personality items.

```
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 S4.

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

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

```r
  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
```

```r
#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.

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

rm(runs_data_2)
```

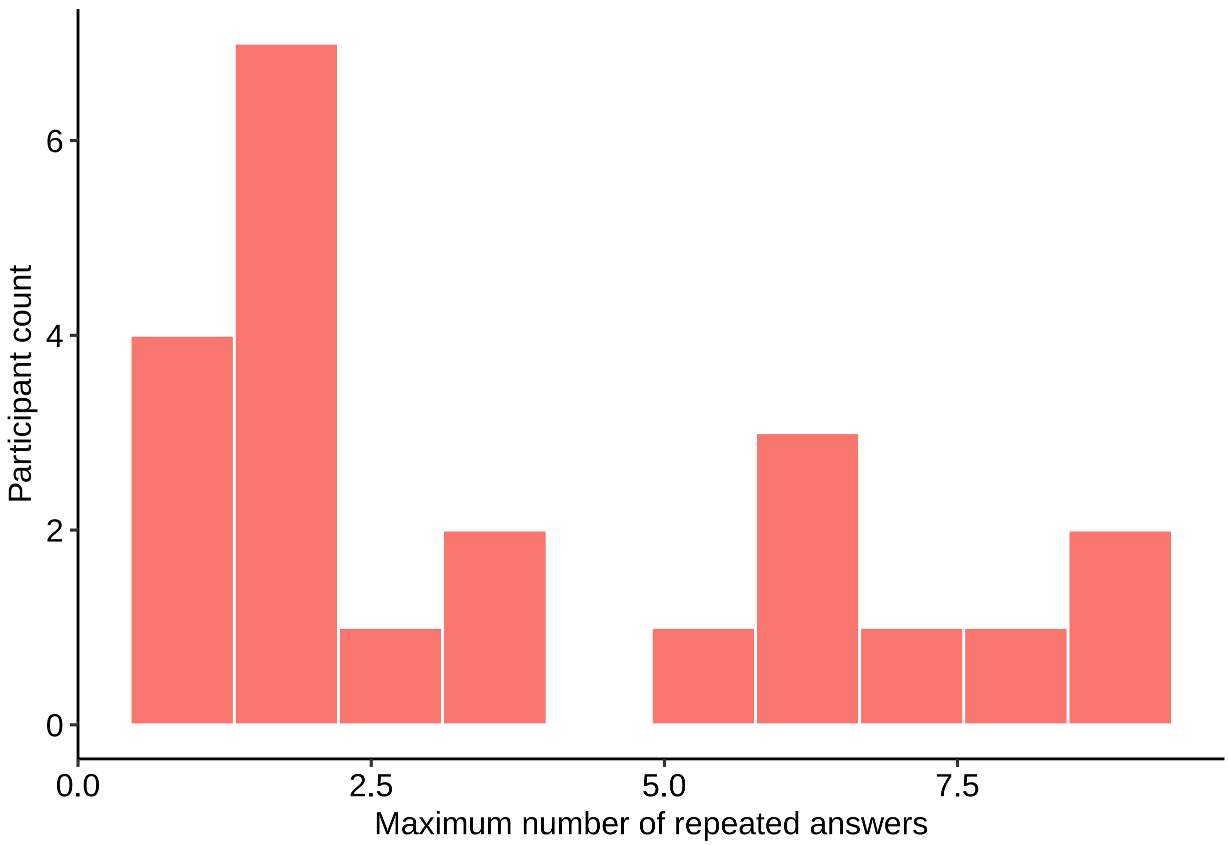


Figure S4: Maximum number of same consecutive responses in personality block 3.

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 S5 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"))
```

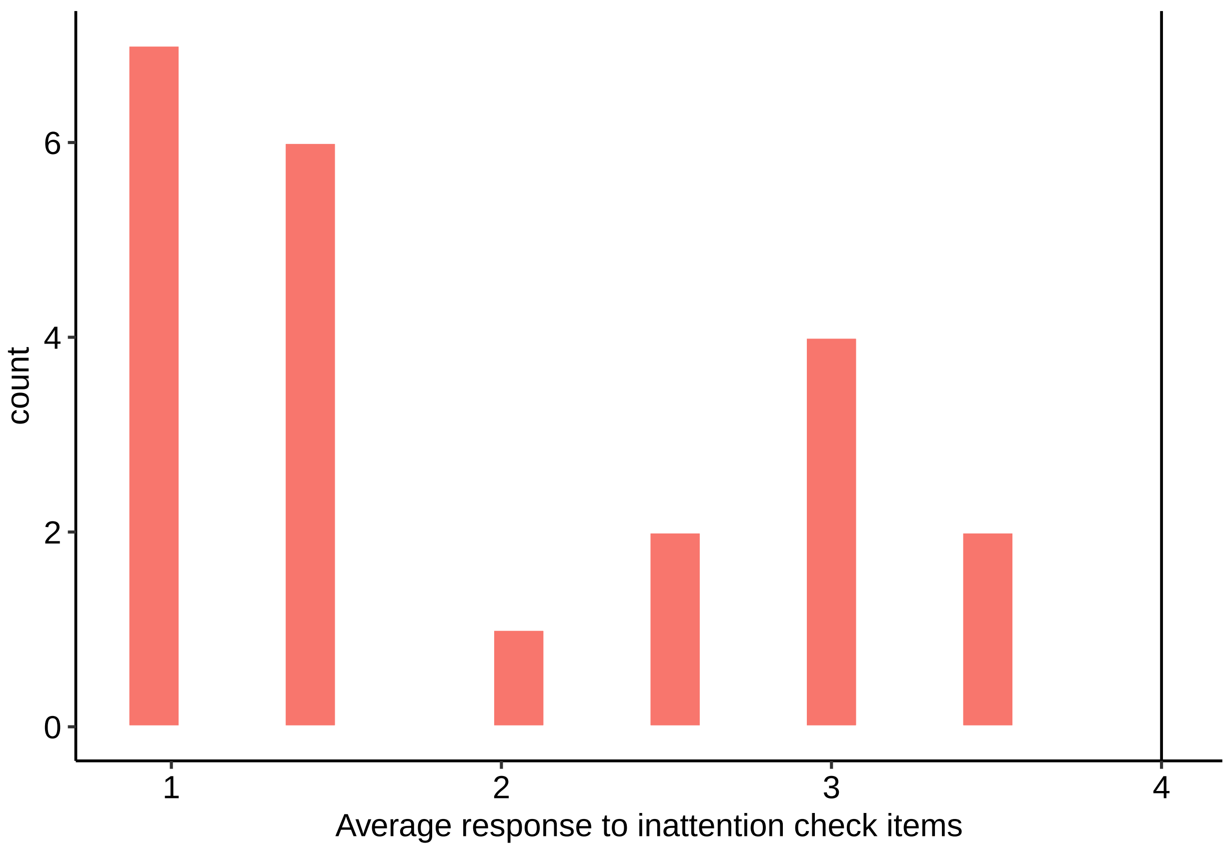


Figure S5: Average response to inattention check items

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 S6 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

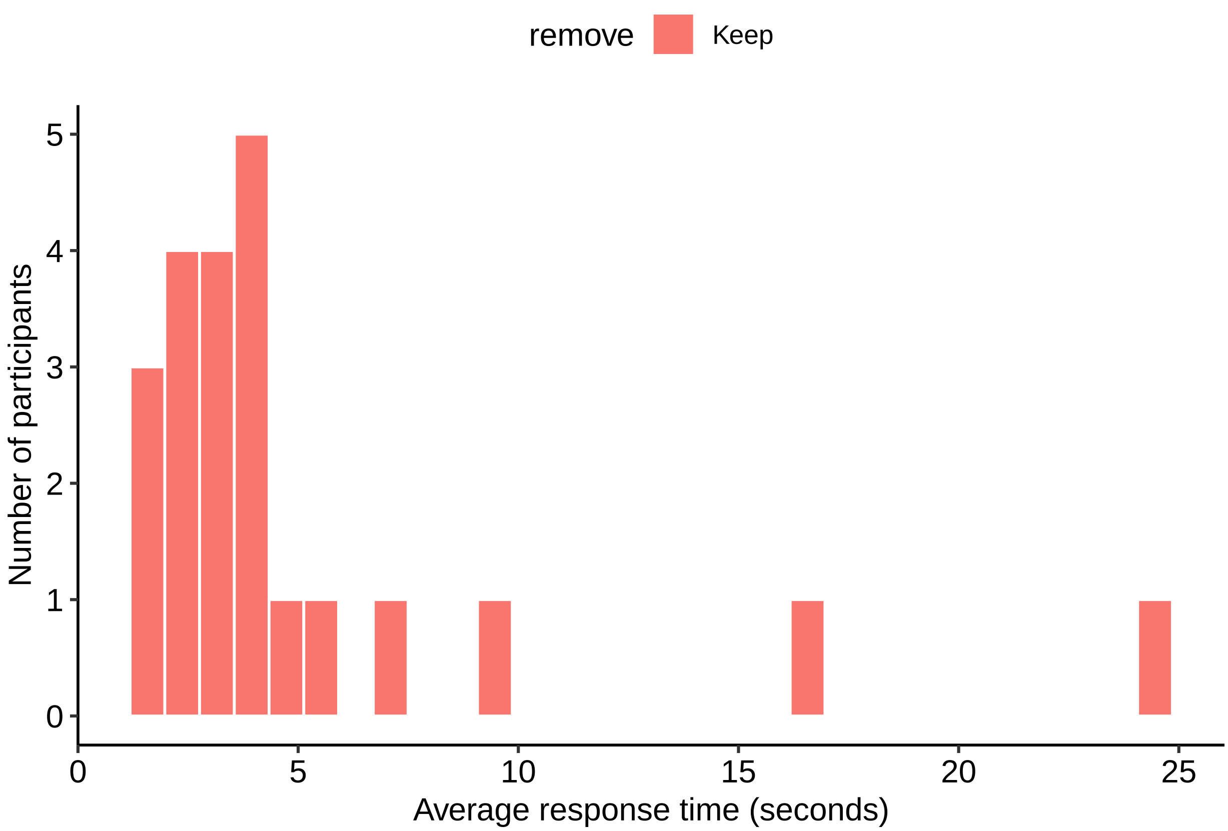


Figure S6: Distribution of average time to respond to personality items in Block 3.

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, child, gold, tree, paper, ring                              6
## 2 tree, skin, river, paper, market, king, hotel, gold, child, book     10
## 3 book, king, market, gold                                          4
```

```
#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 butter, college, earth, ocean, sky, wife                   6
## 3 butter, , earth, , flag, , machine, home, wife, , sky.     7
```

```
#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 blood, corner, woman, rock, shoes, girl, valley                          7
## 2 blood, corner.girl.woman, house, shoes                                   3
## 3 blood, corner, engine, fire, house, letter, shoes, valley, woman, scent      8
```

Table S1: 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 baby, church, water, fire, garden                         5
## 2 water, baby, church, garden, fire                         5
## 3 baby, church, fire, garden, palace, sea, table, water     8
```

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 S7 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
```

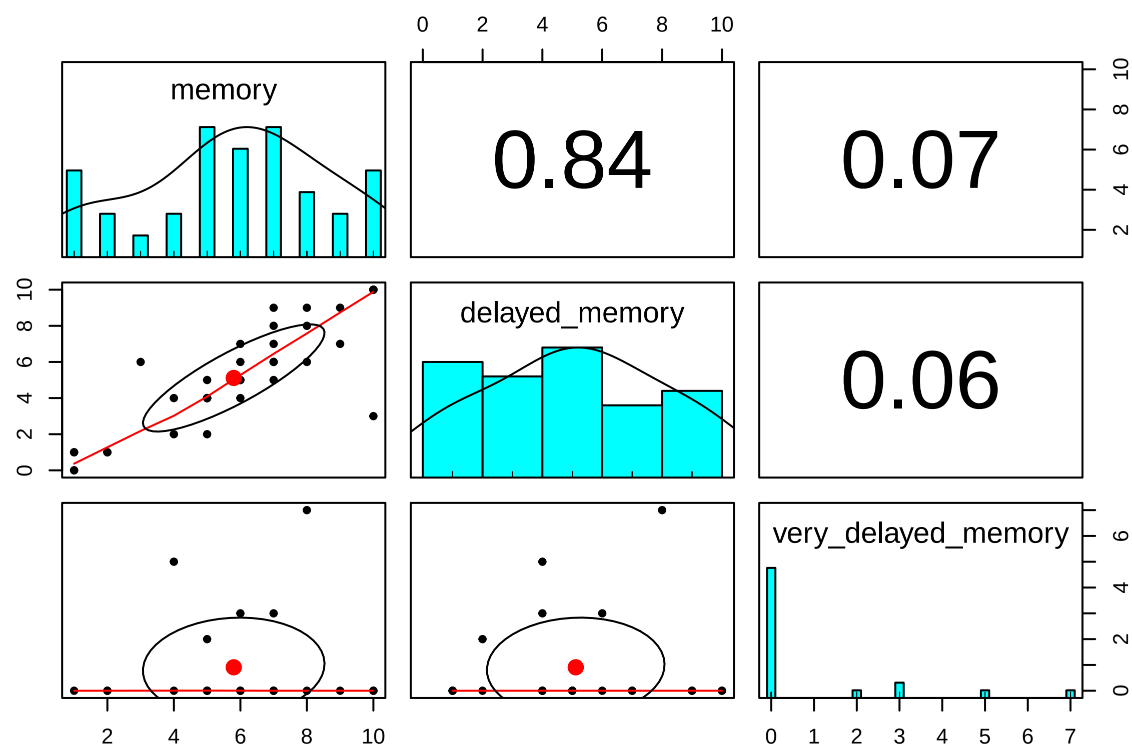


Figure S7: Distributions of memory scores across booth time points.

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 S8). We log-transform this variable for later analyses.

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

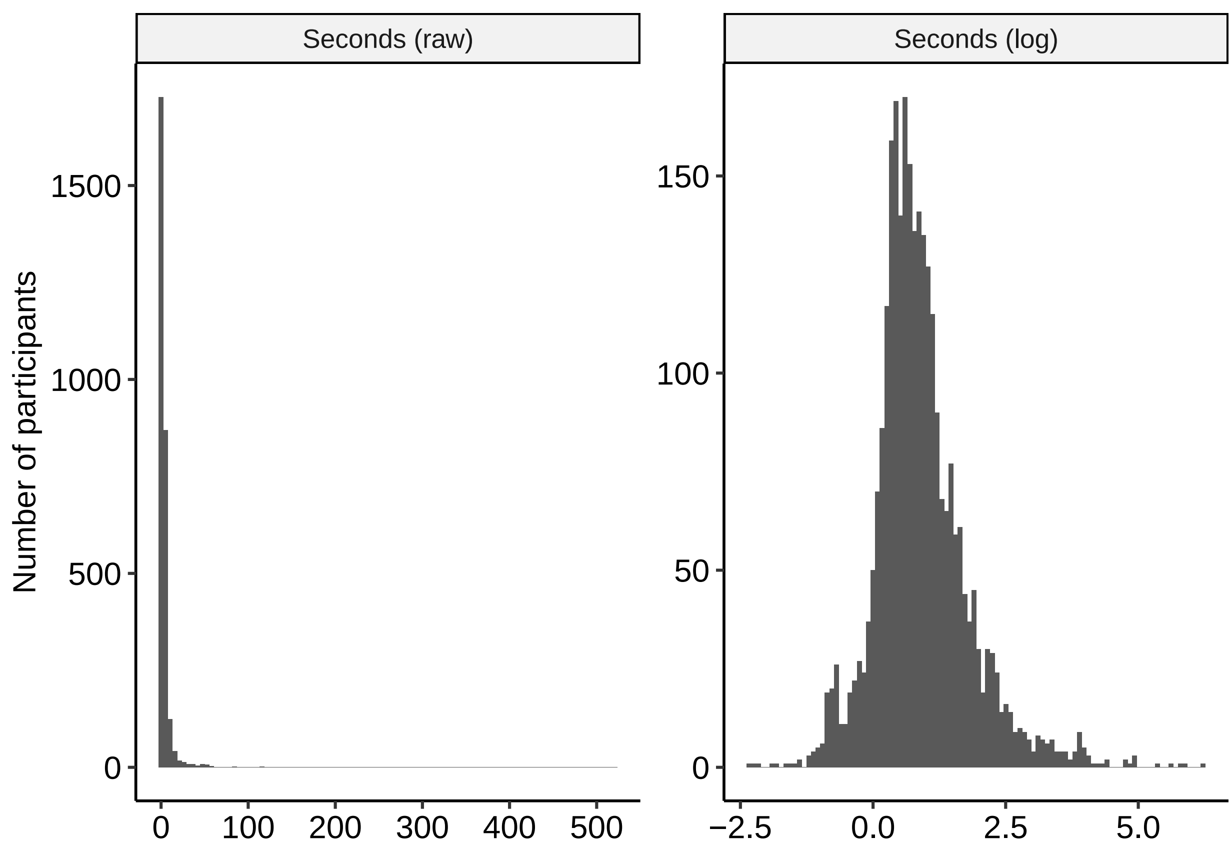


Figure S8: Distribution of seconds, raw and transformed.

2 Descriptives

Participants (N = 35; 22.86% female) were, on average, 39.40 years old (SD = 6.02, minimum = 31, maximum = 51; see Figure S9A for the full distribution). A majority (65.71%) of participants identified as White only (25.71% only); Figure S9B shows the other response options and frequencies. See Figure S9C for the distribution of education, and S9D 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 S2 for the descriptive statistics of each format by block.

See Table S3 for the descriptive statistics of each item and format in Block 1 (Time 1).

See Table S4 for the descriptive statistics of each item and format in Block 2 (Time 1).

See Table S5 for the descriptive statistics of each item and format in Block 3 (Time 2).

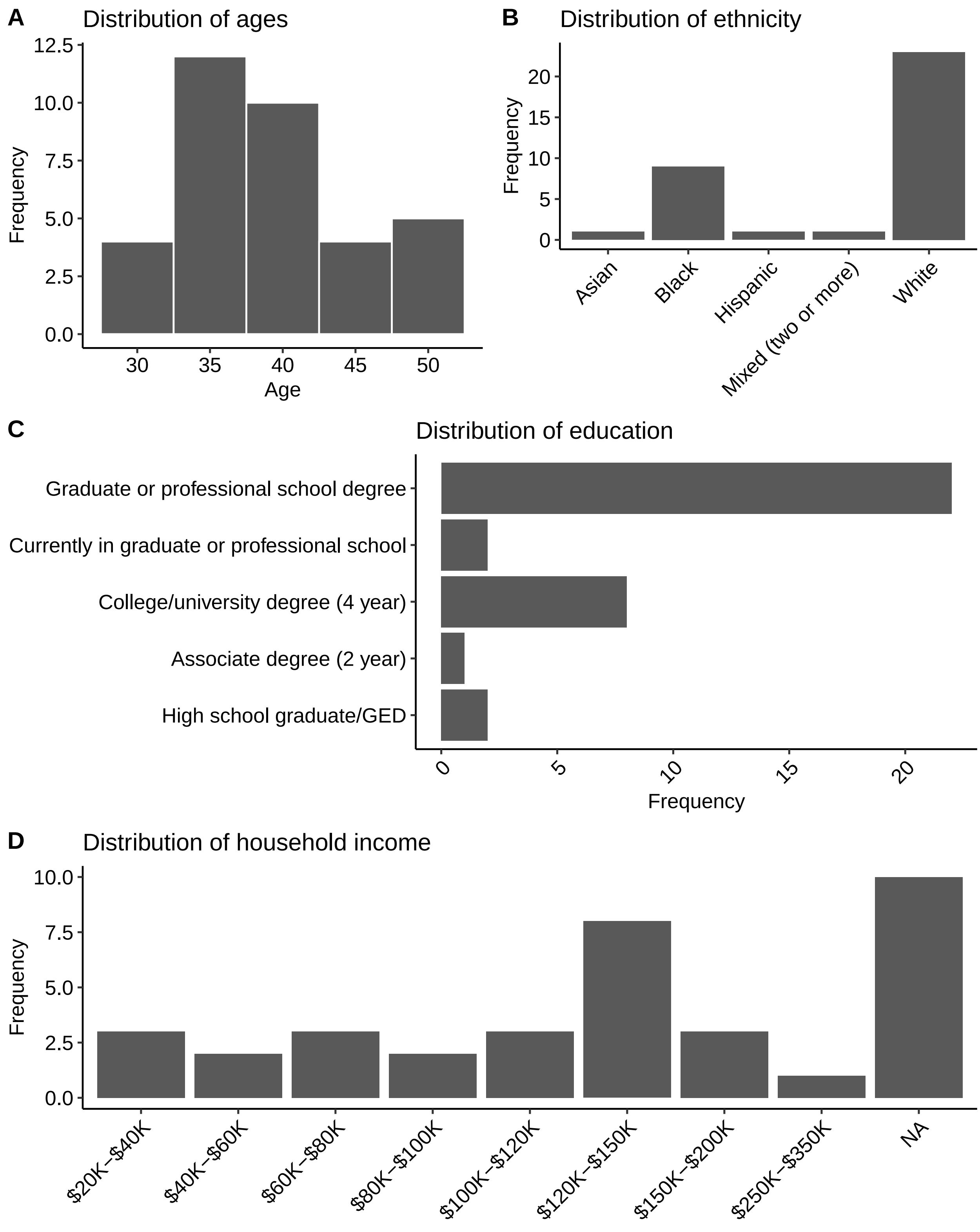


Figure S9: Distributions of key demographics across the entire sample

Table S2: 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 style?

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 participant response style. 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. . .).

In this section, we test the impact of item format on three components of response style:

1. Expected (average) response
2. Likelihood of extreme responding
3. Nay-saying

For these analyses, we use data from Blocks 1 and 2.

3.1 Expected response

We used a multilevel model. Our primary predictor was format. We use data from all three blocks; as a consequence, each person contributes either two or three data points for each of the trait descriptive adjectives. Thus, we nest responses within participant to account for this dependency. This is equivalent to a repeated measures ANOVA. However, in this omnibus model, we include responses to all trait adjectives. Thus, we must also account for adjective-specific contributions to variability. Finally, we include a random term for block. This is not hypothesized to account for significant variability, but we include this term in the event that block contributes significantly to ratings.

We use the `aov` function to calculate the amount of variability in response due to format.

```
mod.expected = items_df %>%
  filter(block %in% c(1,2)) %>%
  glmmTMB(response~format + (1|item) + (1|proid) + (1|block),
                  data = .)

tidy(aov(mod.expected))
```

Table S3: 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 S4: 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 S5: 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)

```
## # A tibble: 5 x 6
##   term         df     sumsq  meansq statistic    p.value
##   <chr>     <dbl>     <dbl>   <dbl>     <dbl>      <dbl>
## 1 format        3    6.97    2.32      2.03    1.08e-  1
## 2 item         30  799.     26.6      23.3     1.13e-108
## 3 proid        34  430.     12.7      11.1     8.17e- 54
## 4 block         1    0.0357  0.0357    0.0312  8.60e-  1
## 5 Residuals  2101 2405.      1.14     NA      NA
```

Item format was unassociated with participants' expected responses to personality items ($F(3.00, 2, 101.00) = 2.03, p = .108$). See Figure S10 for a visualization of this effect. In addition, Figure S11 shows the full distribution of responses across format.

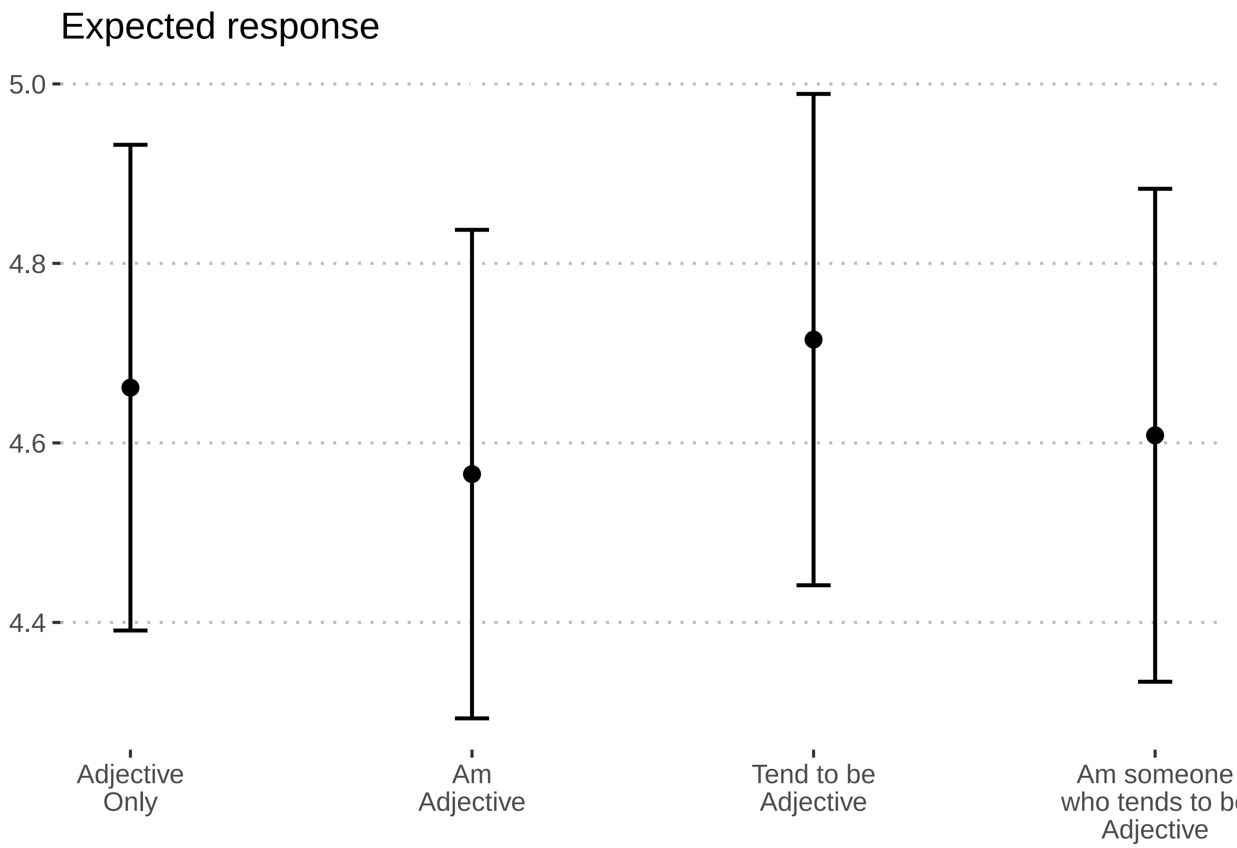


Figure S10: Predicted response on personality items by condition.

3.1.1 One model for each adjective

We repeat this analysis separately for each trait.

```
mod_by_item = items_df %>%
  filter(block %in% c(1,2)) %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~glmmTMB(response~format + (1|proid) + (1|block),
                                  data = .))) %>%
  mutate(aov = map(mod, aov))
```

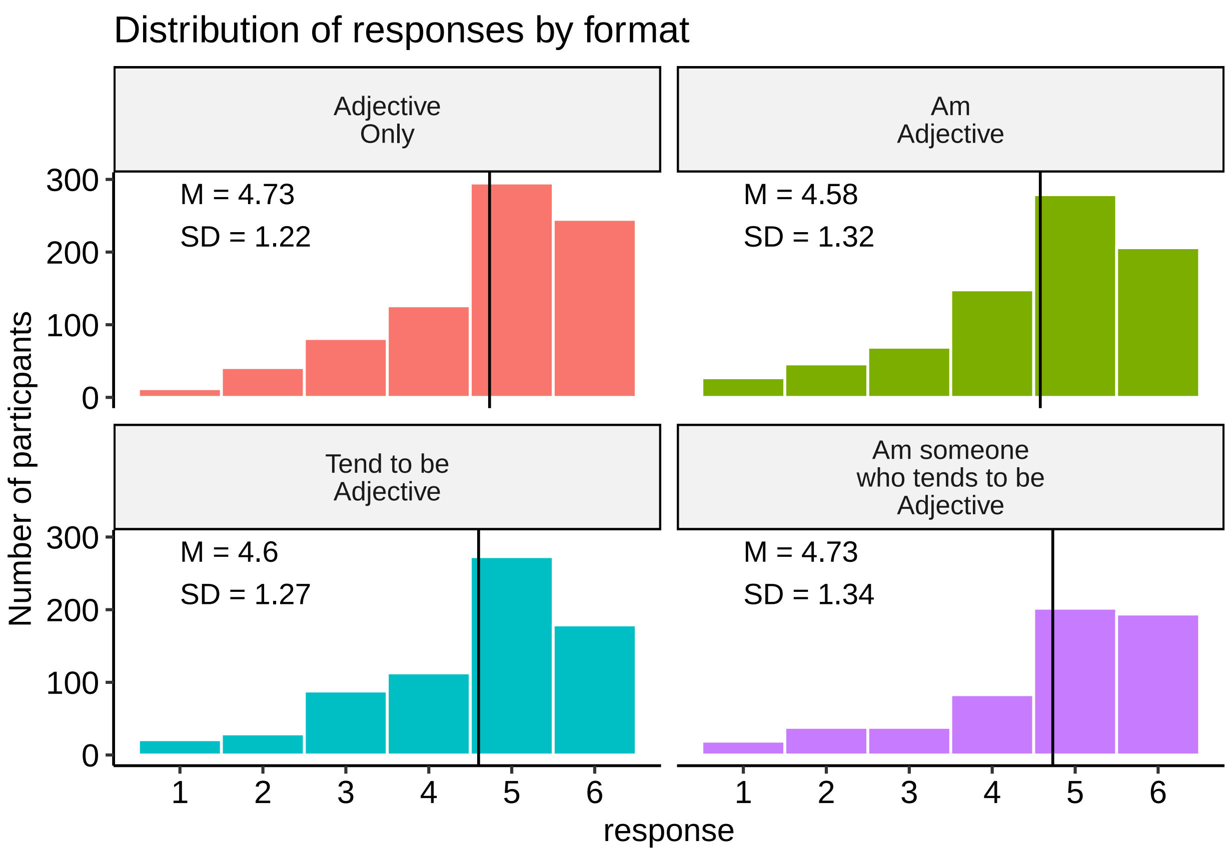


Figure S11: Distribution of responses by category.

Table S6: Format effects on expected response by item.

Item	Reverse Scored?	SS	MS	df1	df2	F	raw	adj
active	N	0.80	0.27	3	31	1.05	.386	> .999
adventurous	N	4.39	1.46	3	31	3.18	.038	.869
broadminded	N	1.59	0.53	3	31	0.76	.524	> .999
calm	N	2.38	0.79	3	31	2.40	.087	> .999
caring	N	1.78	0.59	3	31	1.90	.151	> .999
cautious	N	1.73	0.58	3	31	0.73	.544	> .999
creative	N	0.07	0.02	3	31	0.07	.977	> .999
curious	N	2.52	0.84	3	31	1.09	.368	> .999
friendly	N	0.98	0.33	3	31	1.59	.213	> .999
hardworking	N	4.45	1.48	3	31	4.39	.011	.296
helpful	N	0.52	0.17	3	31	0.49	.690	> .999
imaginative	N	2.85	0.95	3	31	2.42	.085	> .999
intelligent	N	2.74	0.91	3	31	1.78	.171	> .999
lively	N	2.77	0.92	3	31	2.25	.102	> .999
organized	N	1.88	0.63	3	31	3.36	.031	.777
outgoing	N	5.02	1.67	3	31	5.93	.003	.077
responsible	N	1.62	0.54	3	31	0.94	.431	> .999
selfdisciplined	N	2.10	0.70	3	31	1.74	.179	> .999
softhearted	N	2.19	0.73	3	31	0.80	.504	> .999
sophisticated	N	2.33	0.78	3	31	1.71	.185	> .999
sympathetic	N	3.52	1.17	3	31	3.65	.023	.603
talkative	N	10.49	3.50	3	31	3.24	.035	.847
thorough	N	1.99	0.66	3	31	2.49	.079	> .999
thrifty	N	9.59	3.20	3	31	4.85	.007	.201
warm	N	1.02	0.34	3	31	0.86	.473	> .999
careless	Y	23.21	7.74	3	31	7.65	.001	.018
impulsive	Y	0.65	0.22	3	31	0.42	.742	> .999
moody	Y	5.54	1.85	3	31	3.17	.038	.869
nervous	Y	3.66	1.22	3	31	1.18	.333	> .999
reckless	Y	7.28	2.43	3	31	2.25	.103	> .999
worrying	Y	13.61	4.54	3	31	4.86	.007	.201

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 S6, which is organized by whether items were reverse-coded prior to analysis.

3.1.2 Pairwise t-tests for significant ANOVAs

When format was a significant predictor of expected 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.

Differences in means and significance are shown in Table S7. These are also plotted in Figure S12.

```
sig_item = summary_by_item %>%
  filter(p.value < .05)

sig_item = sig_item$item
sig_item
```

```
##  [1] "adventurous" "careless"    "hardworking" "moody"       "organized"
##  [6] "outgoing"    "sympathetic" "talkative"   "thrifty"     "worrying"
```

```
pairwise_response = mod_by_item %>%
  #only significant items
  filter(item %in% sig_item) %>%
  #use emmeans package to calculate format means and run pairwise comparisons
  mutate(comp = map(mod, emmeans, pairwise~format, adjust = "none"),
         means = map(comp, "emmeans"),
         means = map(means, as.data.frame),
         contrasts = map(comp, "contrasts"),
         contrasts = map(contrasts, as.data.frame))
```

```
pairwise_response %>%
  select(item, contrasts) %>%
  unnest(cols = c(contrasts)) %>%
  mutate(estimate = printnum(estimate),
         estimate = case_when(
           p.value < .001 ~ paste0(estimate, "***"),
           p.value < .01 ~ paste0(estimate, "**"),
           p.value < .05 ~ paste0(estimate, "*"),
           TRUE ~ estimate
         )) %>%
  mutate(
    contrast = str_replace(contrast, "Adjective\nOnly", "A"),
    contrast = str_replace(contrast, "Am\nAdjective", "B"),
    contrast = str_replace(contrast, "Tend to be\nAdjective", "C"),
    contrast = str_replace(contrast, "Am someone\nwho tends to be\nAdjective", "D"),
    contrast = str_remove_all(contrast, " ")
  ) %>%
  select(item, contrast, estimate) %>%
  pivot_wider(names_from = contrast, values_from = estimate) %>%
  kable(booktabs = T,
        escape = F,
        caption = "Pairwise differences of means by format. A = Adjective only. B = Am Adjective. C = Te
  kable_styling()
```

```
pairwise_response %>%
  select(item, means) %>%
  unnest(cols = c(means)) %>%
  mutate(format = case_when(
    format == "Adjective\nOnly" ~ 1,
    format == "Am\nAdjective" ~ 2,
    format == "Tend to be\nAdjective" ~ 3,
    format == "Am someone\nwho tends to be\nAdjective" ~ 4)) %>%
```

Table S7: Pairwise differences of means by format. A = Adjective only. B = Am Adjective. C = Tend to be Adjective. D = Am someone who tends to be Adjective. * p < .05, ** p < .01, *** p < .001

item	A-B	A-C	A-D	B-C	B-D	C-D
adventurous	-0.31	-0.19	0.19	0.12	0.49	0.38
careless	0.81	0.14	-0.44	-0.68	-1.26**	-0.58
hardworking	0.29	-0.29	0.09	-0.58*	-0.20	0.38
moody	0.33	-0.07	-0.13	-0.40	-0.46	-0.06
organized	0.22	0.17	0.14	-0.05	-0.09	-0.04
outgoing	0.07	0.35	0.08	0.28	0.00	-0.27
sympathetic	0.29	0.22	0.12	-0.07	-0.17	-0.10
talkative	0.36	-0.60	0.14	-0.96*	-0.22	0.74
thrifty	-0.01	-0.92**	-0.09	-0.91*	-0.08	0.83*
worrying	0.52	0.14	0.34	-0.38	-0.18	0.20

```
  ggplot(aes(x = format, y = emmean)) +
  geom_point(stat = "identity") +
  geom_line(alpha = .3) +
  geom_errorbar(aes(ymin = lower.CL, ymax = upper.CL), width = .3) +
  scale_x_continuous(breaks = c(1:4), labels= c("A","B","C","D")) +
  labs(x = NULL, y = "Expected response") +
  facet_wrap(~item) +
  theme_pubr()
```

3.2 Extreme responding

We define *extreme responding* as answering either a 1 (Very inaccurate) or a 6 (Very accurate). To model likelihood of extreme responding by format, we use logistic regression.

```
items_df = items_df %>%
  mutate(extreme = case_when(
    response == 1 ~ 1,
    response == 6 ~ 1,
    TRUE ~ 0
  ))
```

```
mod.extreme = items_df %>%
  filter(block %in% c(1,2)) %>%
  glmmTMB(extreme~format + (1|proid) + (1|item) + (1|block),
          data = .,
          family = "binomial")
tidy(aov(mod.extreme))
```

```
## # A tibble: 5 x 6
##   term         df   sumsq meansq statistic    p.value
##   <chr>     <dbl>   <dbl>  <dbl>     <dbl>      <dbl>
## 1 format        3   0.208 0.0693     0.432  7.30e-  1
## 2 proid        34 108.    3.17      19.7    5.27e-102
## 3 item         30  21.5   0.717      4.47   1.43e- 14
```

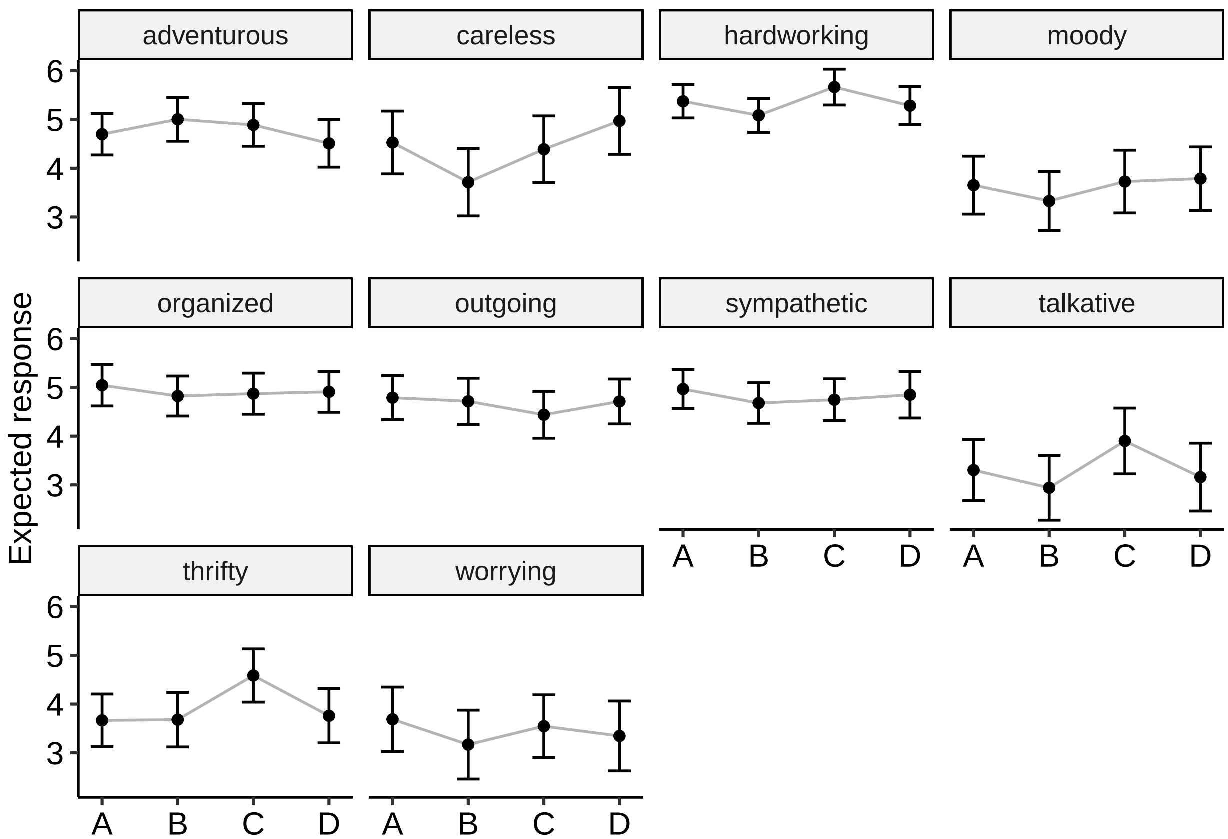


Figure S12: Expected means by format and item. These items were significantly affected by response. A = Adjective only. B = Am Adjective. C = Tend to be Adjective. D = Am someone who tends to be Adjective.

```
## 4 block         1    0.295 0.295      1.84    1.75e-  1
## 5 Residuals  2101 337.    0.160     NA      NA
```

Item format was unassociated with participants' expected responses to personality items ($F(3.00, 2, 101.00) = 0.43, p = .730$). See Figure S13 for a visualization of this effect.

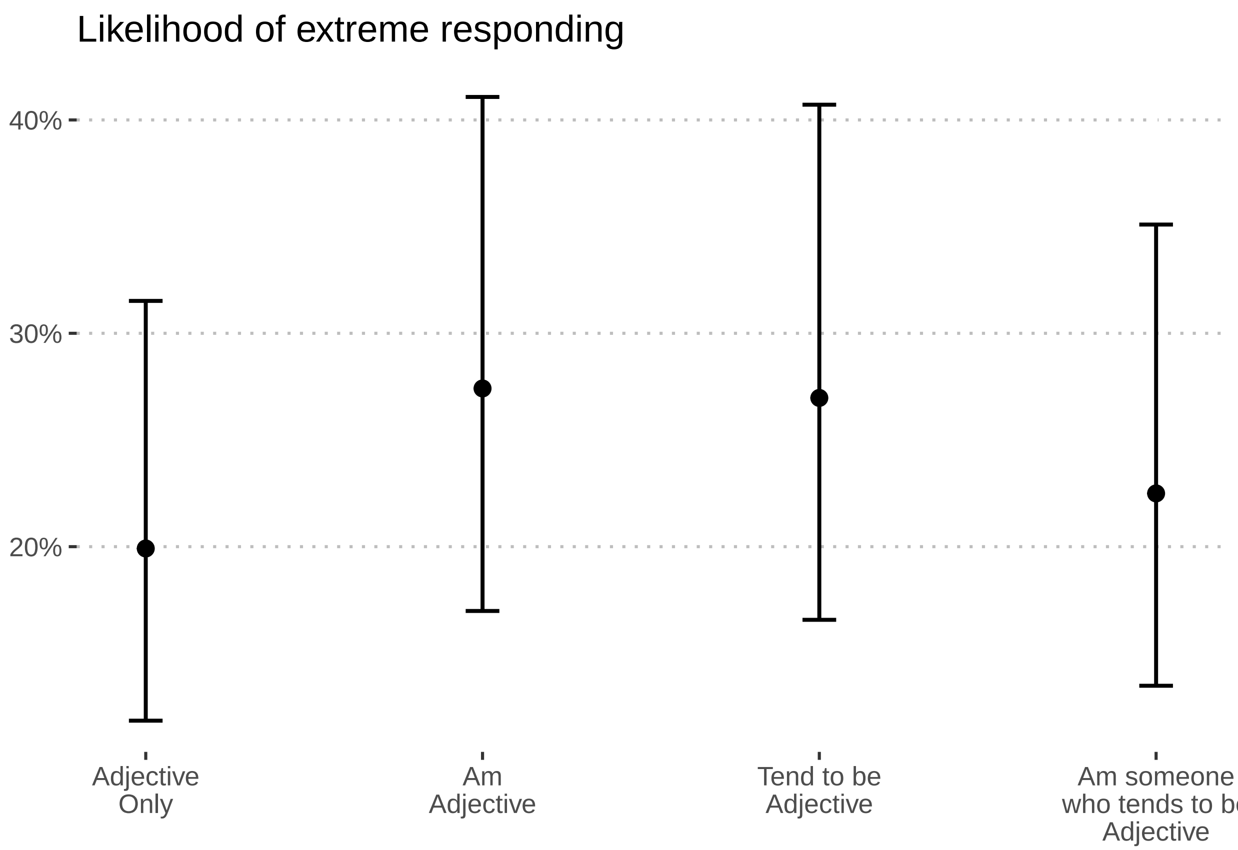


Figure S13: Predicted response on personality items by condition.

3.2.1 One model for each adjective

We repeat this analysis separately for each trait.

```
mod_by_item_ex = items_df %>%
  filter(block %in% c(1,2)) %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~glmmTMB(extreme~format + (1|proid) + (1|block),
                                  data = .,
                                  family = "binomial"))) %>%
  mutate(aov = map(mod, aov))
```

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 S8, which is organized by whether items were reverse-coded prior to analysis.

Table S8: Format effects on extrene response by item.

Item	Reverse Scored?	SS	MS	df	df2	F	raw	adj
active	N	1.04	0.35	3	31	2.97	.047	$> .999$
adventurous	N	0.07	0.02	3	31	0.16	.922	$> .999$
broadminded	N	0.92	0.31	3	31	2.09	.122	$> .999$
calm	N	0.14	0.05	3	31	0.33	.803	$> .999$
caring	N	0.67	0.22	3	31	1.50	.234	$> .999$
cautious	N	0.43	0.14	3	31	1.04	.388	$> .999$
creative	N	0.68	0.23	3	31	2.78	.057	$> .999$
curious	N	0.05	0.02	3	31	0.11	.951	$> .999$
friendly	N	0.48	0.16	3	31	1.34	.280	$> .999$
hardworking	N	2.76	0.92	3	31	9.33	$< .001$	.005
helpful	N	0.90	0.30	3	31	2.78	.058	$> .999$
imaginative	N	0.77	0.26	3	31	1.69	.189	$> .999$
intelligent	N	0.07	0.02	3	31	0.16	.923	$> .999$
lively	N	0.76	0.25	3	31	1.50	.233	$> .999$
organized	N	0.66	0.22	3	31	3.63	.024	.687
outgoing	N	0.13	0.04	3	31	0.68	.574	$> .999$
responsible	N	0.09	0.03	3	31	0.21	.888	$> .999$
selfdisciplined	N	0.65	0.22	3	31	2.15	.114	$> .999$
softhearted	N	0.39	0.13	3	31	1.05	.383	$> .999$
sophisticated	N	0.43	0.14	3	31	1.30	.292	$> .999$
sympathetic	N	0.59	0.20	3	31	2.34	.092	$> .999$
talkative	N	1.63	0.54	3	31	6.89	.001	.033
thorough	N	0.16	0.05	3	31	1.17	.336	$> .999$
thrifty	N	0.41	0.14	3	31	2.02	.132	$> .999$
warm	N	0.59	0.20	3	31	1.27	.302	$> .999$
careless	Y	0.46	0.15	3	31	1.11	.359	$> .999$
impulsive	Y	0.24	0.08	3	31	1.71	.185	$> .999$
moody	Y	0.36	0.12	3	31	0.91	.449	$> .999$
nervous	Y	0.19	0.06	3	31	0.78	.514	$> .999$
reckless	Y	0.39	0.13	3	31	0.90	.450	$> .999$
worrying	Y	0.85	0.28	3	31	2.66	.065	$> .999$

3.2.2 Pairwise t-tests for significant ANOVAs

When format was a significant predictor of extreme responding 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_ex = summary_by_item_ex %>%
  filter(p.value < .05)

sig_item_ex = sig_item_ex$item
sig_item_ex
```

```
## [1] "active"      "hardworking" "organized"   "talkative"
```

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. Likelihood differences are shown in Table S9 and likelihood estimates are in Figure S14.

```
pairwise_response_ex = mod_by_item_ex %>%
  #only significant items
  filter(item %in% sig_item_ex) %>%
  #use emmeans package to calculate format means and run pairwise comparisons
  mutate(comp = map(mod, emmeans, pairwise~format, adjust = "none"),
         means = map(comp, "emmeans"),
         means = map(means, as.data.frame),
         contrasts = map(comp, "contrasts"),
         contrasts = map(contrasts, as.data.frame))
```

```
pairwise_response_ex %>%
  select(item, contrasts) %>%
  unnest(cols = c(contrasts)) %>%
  mutate(estimate = printnum(estimate),
         estimate = case_when(
           p.value < .001 ~ paste0(estimate, "***"),
           p.value < .01 ~ paste0(estimate, "**"),
           p.value < .05 ~ paste0(estimate, "*"),
           TRUE ~ estimate
         )) %>%
  mutate(
    contrast = str_replace(contrast, "Adjective\nOnly", "A"),
    contrast = str_replace(contrast, "Am\nAdjective", "B"),
    contrast = str_replace(contrast, "Tend to be\nAdjective", "C"),
    contrast = str_replace(contrast, "Am someone\nwho tends to be\nAdjective", "D"),
    contrast = str_remove_all(contrast, " ")
  ) %>%
  select(item, contrast, estimate) %>%
  pivot_wider(names_from = contrast, values_from = estimate) %>%
  kable(booktabs = T,
        escape = F,
        caption = "Pairwise differences in likelihood of extreme responding by format. A = Adjective onl
  kable_styling()
```

Table S9: Pairwise differences in likelihood of extreme responding by format. A = Adjective only. B = Am Adjective. C = Tend to be Adjective. D = Am someone who tends to be Adjective. * p < .05, ** p < .01, *** p < .001

item	A-B	A-C	A-D	B-C	B-D	C-D
active	1.31	2.50	2.41	1.19	1.10	-0.09
hardworking	0.22	-18.51***	8.74	-18.73***	8.52	27.25**
organized	0.28	-1.14	7.69	-1.42	7.40	8.82
talkative	-39.52	-52.17	-25.82	-12.65	13.70	26.35

```r
logit2prob <- function(logit){
  odds <- exp(logit)
  prob <- odds / (1 + odds)
  return(prob)
}

pairwise_response_ex %>%
  select(item, means) %>%
  unnest(cols = c(means)) %>%
  mutate(format = case_when(
    format == "Adjective\nOnly" ~ 1,
    format == "Am\nAdjective" ~ 2,
    format == "Tend to be\nAdjective" ~ 3,
    format == "Am someone\nwho tends to be\nAdjective" ~ 4)) %>%
  mutate(across(c(emmean, lower.CL, upper.CL), logit2prob)) %>%
  ggplot(aes(x = format, y = emmean)) +
  geom_point(stat = "identity") +
  geom_line(alpha = .3) +
  geom_errorbar(aes(ymin = lower.CL, ymax = upper.CL), width = .3) +
  scale_x_continuous(breaks = c(1:4), labels= c("A","B","C","D")) +
  labs(x = NULL, y = "Probability of extreme response") +
  facet_wrap(~item) +
  theme_pubr()
```

3.3 Yea-saying

We define *yea-saying* as answering "somewhat accurate" (4), "accurate" (5), or "very accurate" (6) to an item. To model likelihood of extreme responding by format, we use logistic regression. As a reminder, we reverse-scored socially desirable items during the cleaning stage. For those items, responses coded as 1, 2, or 3 represent agreement (accurate). Therefore, we code values 1, 2, and 3 as yea-saying for reverse-scored items, and values 4, 5, and 6 as yea-saying for all other items.

```r
items_df = items_df %>%
  mutate(
    yeasaying = case_when(
    item %in% reverse & response %in% c(1:3) ~ 1,
    !(item %in% reverse) & response %in% c(4:6) ~ 1,
    TRUE ~ 0
  ))
```

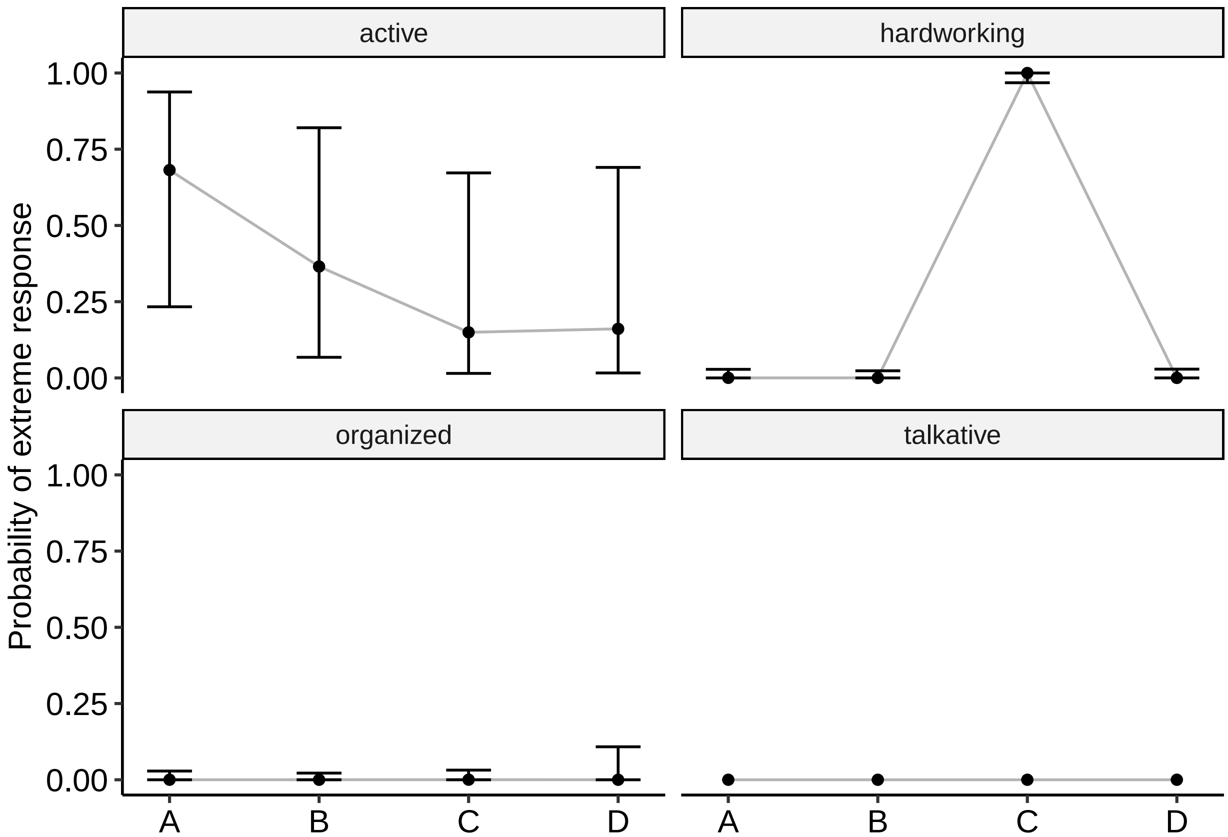


Figure S14: Extreme responding by format and item. These items were significantly affected by response. A = Adjective only. B = Am Adjective. C = Tend to be Adjective. D = Am someone who tends to be Adjective.

```r
mod.yeasaying = items_df %>%
  filter(block %in% c(1,2)) %>%
  glmmTMB(yeasaying~format + (1|proid) + (1|item) + (1|block),
                  data = .,
                  family = "binomial")
tidy(aov(mod.yeasaying))
```

```
## # A tibble: 5 x 6
##   term         df     sumsq  meansq statistic    p.value
##   <chr>     <dbl>     <dbl>   <dbl>     <dbl>      <dbl>
## 1 format        3   0.254   0.0846     0.773   5.09e-  1
## 2 proid        34  28.2     0.828      7.57    9.95e- 34
## 3 item         30 105.      3.50      32.0     7.20e-148
## 4 block         1   0.00513 0.00513    0.0469  8.28e-  1
## 5 Residuals  2101 230.      0.109     NA      NA
```

Item format was unassociated with participants' expected responses to personality items ($F(3.00, 2, 101.00) = 0.77, p = .509$). See Figure S15 for a visualization of this effect.

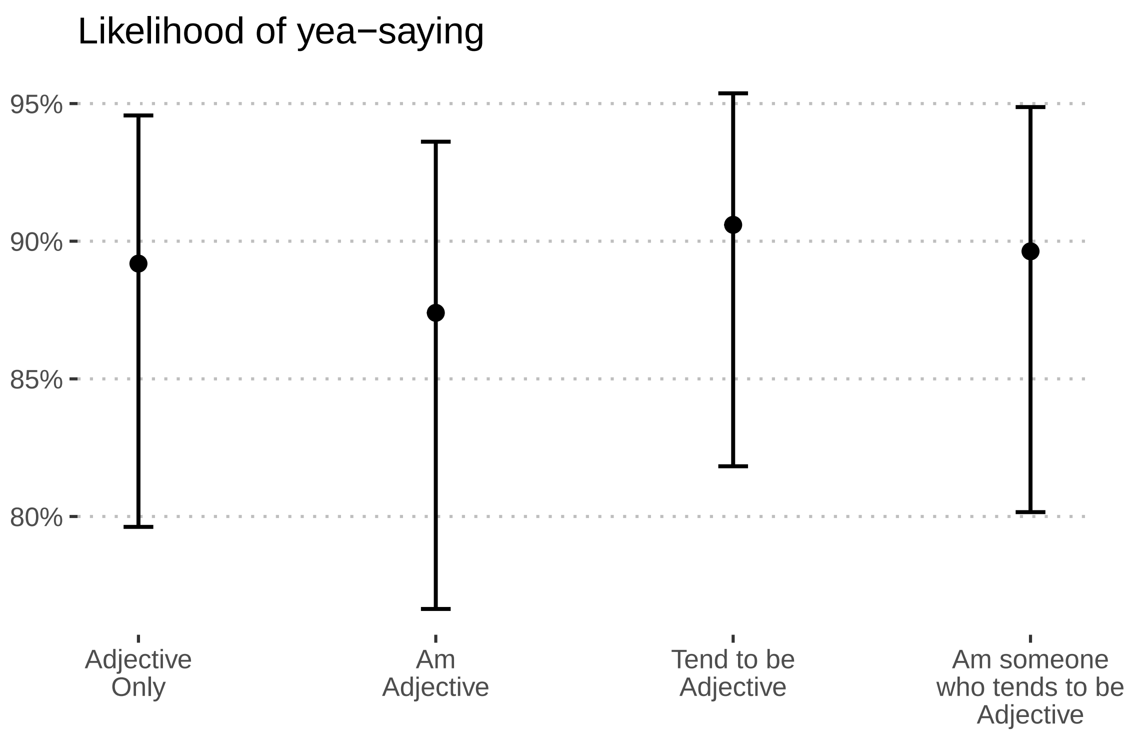


Figure S15: Predicted response on personality items by condition.

3.3.1 One model for each adjective

We repeat this analysis separately for each trait.

```r
mod_by_item_ya = items_df %>%
  group_by(item) %>%
  nest() %>%
```

Table S10: Format effects on yea-saying by item.

Item	Reverse Scored?	SS	MS	df	df2	F	raw	adj
active	N	0.17	0.06	3	52	2.80	.049	$>$ .999
adventurous	N	0.12	0.04	3	52	1.84	.151	$>$ .999
broadminded	N	0.10	0.03	3	52	0.76	.523	$>$ .999
calm	N	0.41	0.14	3	52	2.77	.051	$>$ .999
caring	N	0.04	0.01	3	52	0.96	.419	$>$ .999
cautious	N	0.30	0.10	3	52	2.15	.105	$>$ .999
creative	N	0.03	0.01	3	52	0.30	.827	$>$ .999
curious	N	0.21	0.07	3	52	0.92	.436	$>$ .999
friendly	N	0.03	0.01	3	52	1.19	.323	$>$ .999
hardworking	N	0.04	0.01	3	52	0.73	.538	$>$ .999
helpful	N	0.05	0.02	3	52	0.77	.514	$>$ .999
imaginative	N	0.04	0.01	3	52	0.58	.629	$>$ .999
intelligent	N	0.15	0.05	3	52	3.20	.031	.857
lively	N	0.19	0.06	3	52	1.41	.252	$>$ .999
organized	N	0.27	0.09	3	52	4.23	.009	.285
outgoing	N	0.68	0.23	3	52	3.20	.031	.857
responsible	N	0.17	0.06	3	52	2.71	.054	$>$ .999
selfdisciplined	N	0.20	0.07	3	52	2.11	.110	$>$ .999
softhearted	N	0.15	0.05	3	52	0.89	.451	$>$ .999
sophisticated	N	0.84	0.28	3	52	2.54	.067	$>$ .999
sympathetic	N	0.01	0.00	3	52	0.11	.954	$>$ .999
talkative	N	0.59	0.20	3	52	1.39	.257	$>$ .999
thorough	N	0.09	0.03	3	52	1.79	.160	$>$ .999
thrifty	N	1.14	0.38	3	52	2.69	.056	$>$ .999
warm	N	0.09	0.03	3	52	1.74	.171	$>$ .999
careless	Y	1.18	0.39	3	52	2.58	.064	$>$ .999
impulsive	Y	0.59	0.20	3	52	1.67	.185	$>$ .999
moody	Y	2.16	0.72	3	52	10.73	$<$ .001	$<$ .001
nervous	Y	1.21	0.40	3	52	2.91	.043	$>$ .999
reckless	Y	1.02	0.34	3	52	4.14	.011	.305
worrying	Y	0.18	0.06	3	52	0.35	.787	$>$ .999

```
mutate(mod = map(data, ~glmmTMB(yeasaying~format + (1|proid) + (1|block),
                               data = .,
                               family = "binomial"))) %>%
mutate(aov = map(mod, aov))
```

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 S10, which is organized by whether items were reverse-coded prior to analysis.

3.3.2 Pairwise t-tests for significant ANOVAs

When format was a significant predictor of extreme responding 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.

```r
sig_item_ya = summary_by_item_ya %>%
  filter(p.value < .05)

sig_item_ya = sig_item_ya$item
sig_item_ya
```

```
## [1] "active"      "intelligent" "moody"       "nervous"     "organized"
## [6] "outgoing"    "reckless"
```

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. Likelihood differences are shown in Table S9 and likelihood estimates are in Figure S14.

```r
pairwise_response_ya = mod_by_item_ya %>%
  #only significant items
  filter(item %in% sig_item_ya) %>%
  #use emmeans package to calculate format means and run pairwise comparisons
  mutate(comp = map(mod, emmeans, pairwise~format, adjust = "none"),
         means = map(comp, "emmeans"),
         means = map(means, as.data.frame),
         contrasts = map(comp, "contrasts"),
         contrasts = map(contrasts, as.data.frame))
```

```r
pairwise_response_ya %>%
  select(item, contrasts) %>%
  unnest(cols = c(contrasts)) %>%
  mutate(estimate = printnum(estimate),
         estimate = case_when(
           p.value < .001 ~ paste0(estimate, "***"),
           p.value < .01 ~ paste0(estimate, "**"),
           p.value < .05 ~ paste0(estimate, "*"),
           TRUE ~ estimate
         )) %>%
  mutate(
    contrast = str_replace(contrast, "Adjective\nOnly", "A"),
    contrast = str_replace(contrast, "Am\nAdjective", "B"),
    contrast = str_replace(contrast, "Tend to be\nAdjective", "C"),
    contrast = str_replace(contrast, "Am someone\nwho tends to be\nAdjective", "D"),
    contrast = str_remove_all(contrast, " ")
  ) %>%
  select(item, contrast, estimate) %>%
  pivot_wider(names_from = contrast, values_from = estimate) %>%
  kable(booktabs = T,
        escape = F,
        caption = "Pairwise differences in likelihood of yea-saying by format. A = Adjective only. B = A
  kable_styling()
```

Table S11: Pairwise differences in likelihood of yea-saying by format. A = Adjective only. B = Am Adjective. C = Tend to be Adjective. D = Am someone who tends to be Adjective. * p < .05, ** p < .01, *** p < .001

item	A-B	A-C	A-D	B-C	B-D	C-D
active	11.96	8.87	6.73	-3.09	-5.24	-2.14
intelligent	0.00	-322.34	-84.30	-322.34	-84.30	238.04
moody	-2.34	-1.50	-0.56	0.85	1.79	0.94
nervous	1.81	-0.57	2.00	-2.38	0.19	2.57*
organized	51.13	50.64	36.15	-0.49	-14.99	-14.49
outgoing	-2.94	0.40	1.34	3.34	4.28	0.94
reckless	9.30	16.77	19.72*	7.47	10.42	2.95

```r
pairwise_response_ya %>%
  select(item, means) %>%
  unnest(cols = c(means)) %>%
  mutate(format = case_when(
    format == "Adjective\nOnly" ~ 1,
    format == "Am\nAdjective" ~ 2,
    format == "Tend to be\nAdjective" ~ 3,
    format == "Am someone\nwho tends to be\nAdjective" ~ 4)) %>%
  mutate(across(c(emmean, lower.CL, upper.CL), logit2prob)) %>%
  ggplot(aes(x = format, y = emmean)) +
  geom_point(stat = "identity") +
  geom_line(alpha = .3) +
  geom_errorbar(aes(ymin = lower.CL, ymax = upper.CL), width = .3) +
  scale_x_continuous(breaks = c(1:4), labels= c("A","B","C","D")) +
  labs(x = NULL, y = "Probability of yeasaying") +
  facet_wrap(~item) +
  theme_pubr()
```

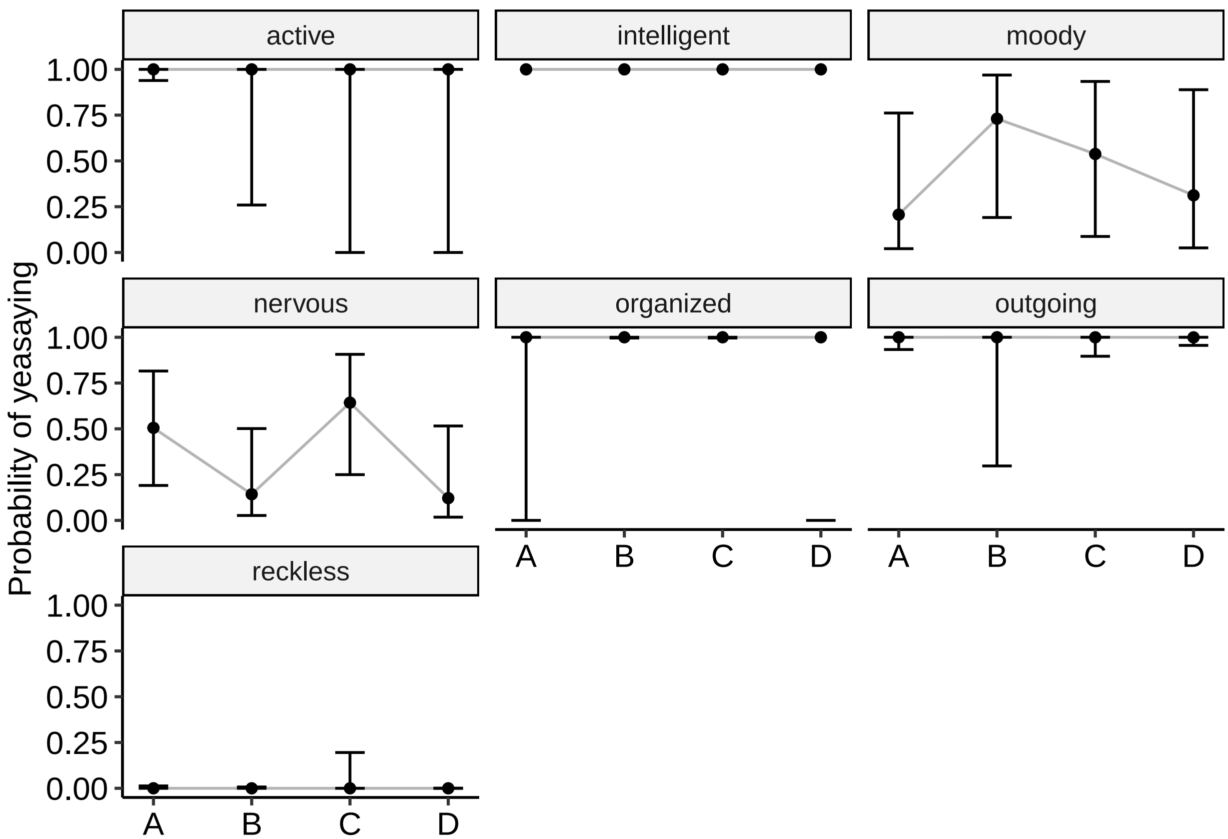
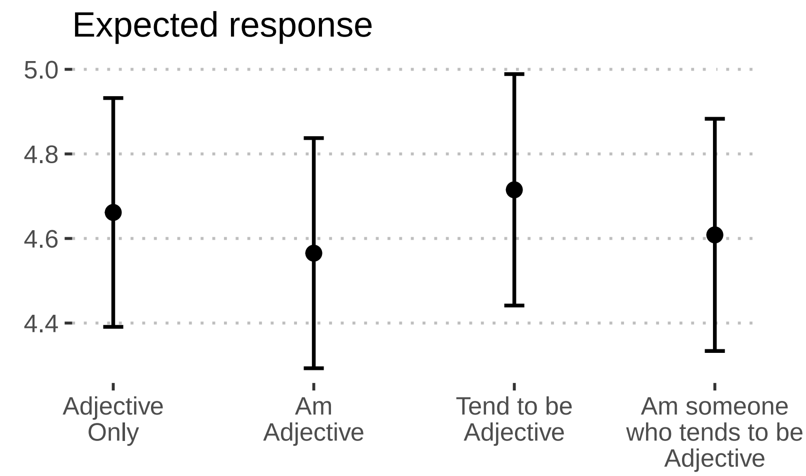


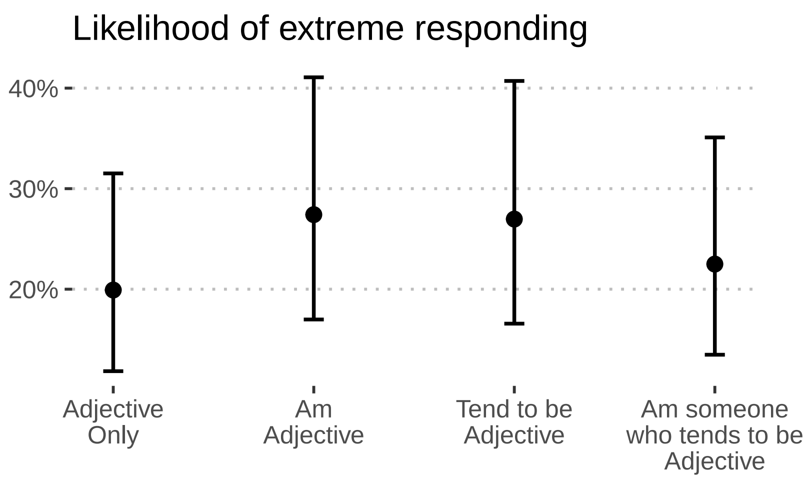
Figure S16: Yea-saying by format and item. These items were significantly affected by response. A = Adjective only. B = Am Adjective. C = Tend to be Adjective. D = Am someone who tends to be Adjective.

3.4 All tests

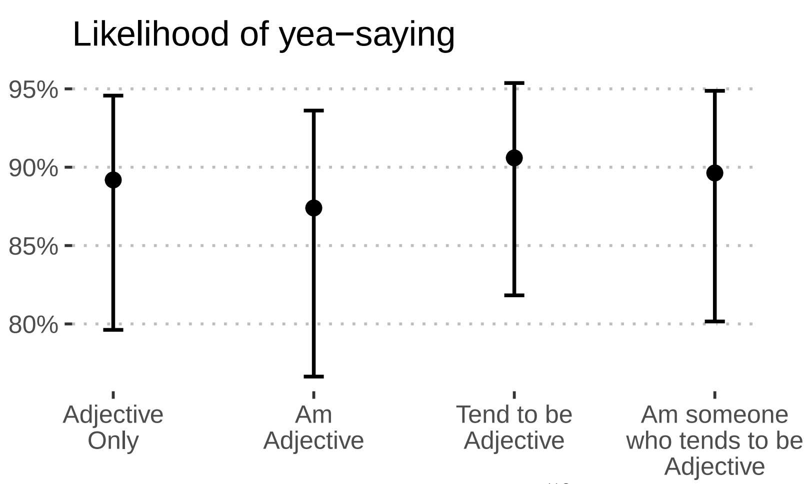
A



B



C



4 Does the internal consistency and reliability of Big Five traits vary by item wording?

We calculate and report Cronbach's alpha for all formats using data from Blocks 1 and 2. This will include both the average split-half reliability, as well as the 95% confidence interval. Differences in internal consistency will be considered statistically significant if the confidence intervals of two formats do not overlap. We will also show the distribution of all possible split halves for each of the four formats.

We start by creating a wide-format of the dataset using only the Block 1 data.

```
items_wide = items_df %>%
  # only blocks 1 and 2
  filter(block %in% c(1,2)) %>%
  #only need these variables
  select(proid, block, condition, item, response) %>%
  # to wide form
  spread(item, response)
```

Next, we identify the items associated with each trait. These come from the Health and Retirement Study Psychosocial and Lifestyle Questionnaire 2006-2016 user guide, which can be found at this link.

```
Extra = c("outgoing", "friendly", "lively", "active" ,"talkative")
Agree = c("helpful", "warm", "caring", "softhearted", "sympathetic")
Consc = c("reckless", "organized", "responsible", "hardworking", "selfdisciplined",
          "careless", "impulsive", "cautious", "thorough", "thrifty")
Neuro = c("moody", "worrying", "nervous", "calm")
Openn = c("creative", "imaginative", "intelligent", "curious", "broadminded",
          "sophisticated", "adventurous")
```

4.1 Calculate Cronbach's alpha for each format

We start by grouping data by condition and then nesting, to create separate data frames for each of the four formats.

```
format_data = items_wide %>%
  group_by(condition) %>%
  nest() %>%
  ungroup()
```

Next we create separate datasets for each of the five personality traits.

```
format_data = format_data %>%
  mutate(
    data_Extra = map(data, ~select(.x, all_of(Extra))),
    data_Agree = map(data, ~select(.x, all_of(Agree))),
    data_Consc = map(data, ~select(.x, all_of(Consc))),
    data_Neuro = map(data, ~select(.x, all_of(Neuro))),
    data_Openn = map(data, ~select(.x, all_of(Openn)))
    )
```

We gather these datasets into a single column, for ease of use.

Table S12: Cronbach's alpha across format and trait.

label	A	B	C	D
Extraversion (5 descriptors)	0.12 [-0.47, 0.70]	0.68 [0.48, 0.88]	0.60 [0.28, 0.93]	0.86 [0.75, 0.97]
Agreeableness (5 descriptors)	0.51 [0.19, 0.82]	0.57 [0.25, 0.89]	0.90 [0.83, 0.97]	0.23 [-0.42, 0.88]
Conscientiousness (10 descriptors)	0.43 [0.09, 0.76]	0.56 [0.28, 0.83]	0.80 [0.65, 0.95]	0.51 [0.11, 0.91]
Neuroticism (4 descriptors)	0.67 [0.47, 0.86]	0.71 [0.54, 0.89]	0.37 [-0.14, 0.88]	0.72 [0.50, 0.93]
Openness (7 descriptors)	0.56 [0.27, 0.84]	0.70 [0.49, 0.90]	0.85 [0.74, 0.96]	0.41 [-0.08, 0.90]

```
format_data = format_data %>%
  select(-data) %>%
  gather(variable, data, starts_with("data")) %>%
  mutate(variable = str_remove(variable, "data_"))
```

Next we apply the `alpha` and `omega` functions to the datasets. We do not need to use the `check.keys` function, as items were reverse-scored during the cleaning process.

```
format_data = format_data %>%
  mutate(
    nvar = map_dbl(data, ncol),
    alpha = map(data, psych::alpha),
    omega = map(data, psych::omega, plot = F))
```

4.2 Alpha

We extract the estimated confidence intervals. (Note that these estimates are unreliable in small samples. The estimates extracted based on pilot data are not expected to reflect estimates provided in the final analyses.) The final summary of results is presented in Table S12 and Figure S17.

```
format_alpha = format_data %>%
  mutate(alpha_list = map(alpha, "total"),
         alpha_est = map_dbl(alpha_list, "raw_alpha"),
         se_est = map_dbl(alpha_list, "ase"),
         lower_est = alpha_est - (1.96*se_est),
         upper_est = alpha_est + (1.96*se_est))
```

4.3 Split-half reliability

Alpha is the average split-half reliability; given space, it can be useful to report the distribution of all split-half reliability estimates. We use the `splitHalf` function to calculate those. We use smoothed correlation matrices here because when developing code on the pilot data, we had non-positive definite correlation matrices. See Figure S18 for these distributions.

```
format_split = format_data %>%
  mutate(cor_mat = map(data, cor),
         cor_mat = map(cor_mat, cor.smooth)) %>%
  mutate(splithalf = map(cor_mat, splitHalf, raw = T))
```

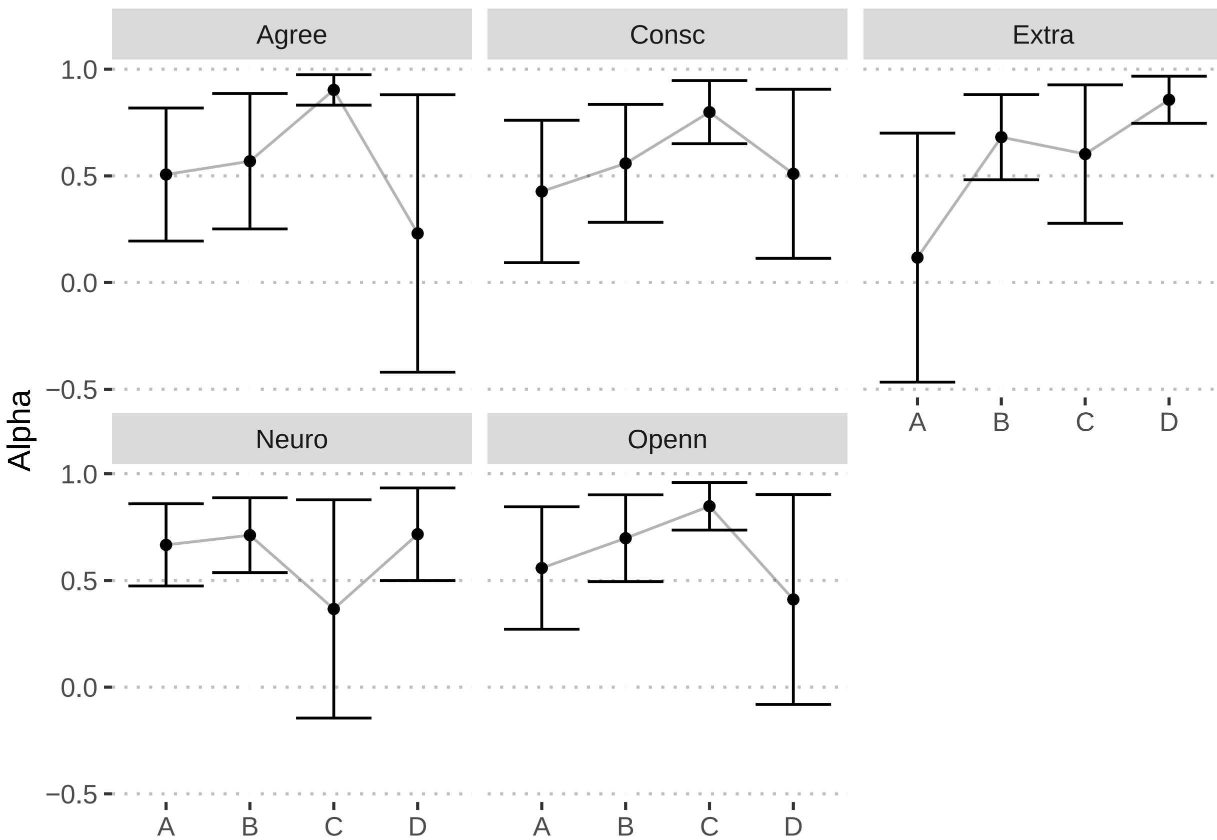


Figure S17: Estimates of Cronbach's alpha across format and trait.

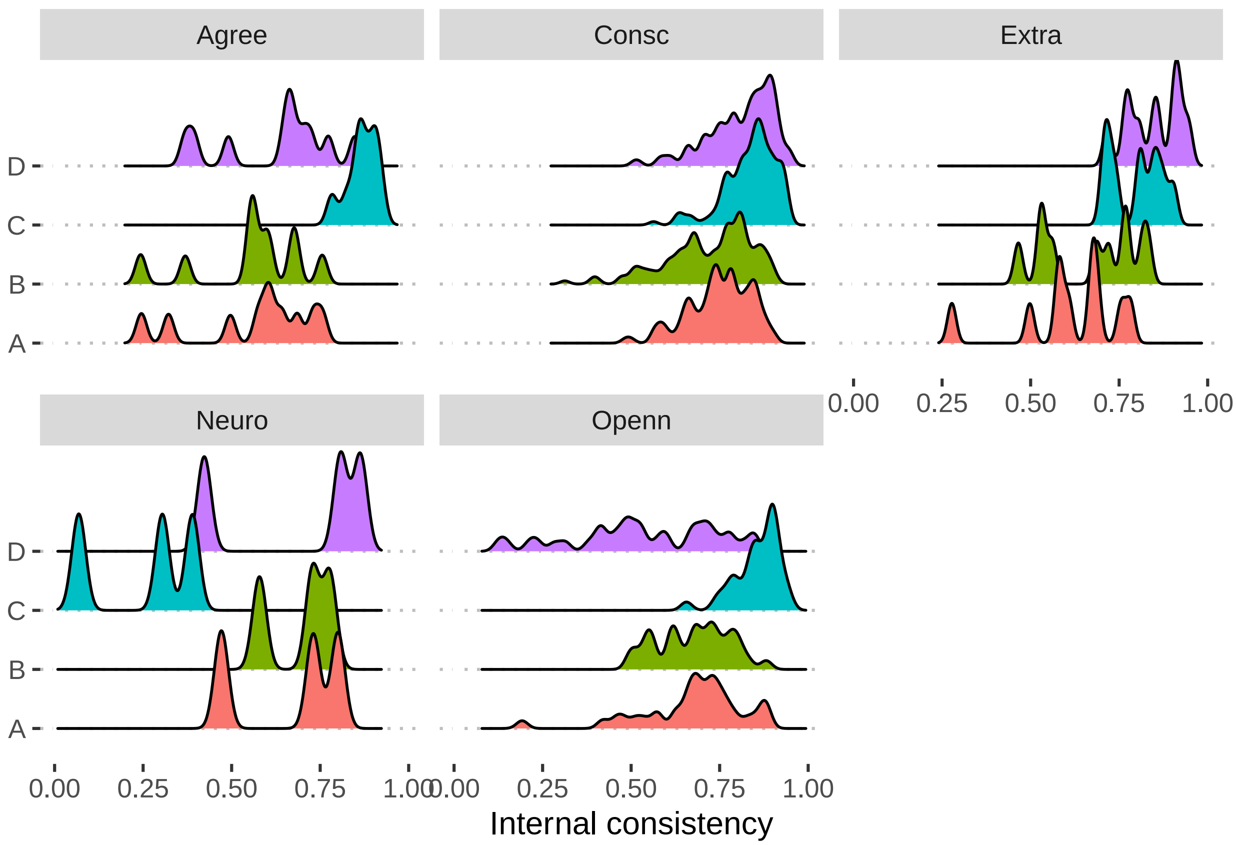


Figure S18: Distribution of split-half reliabilities

Table S13: Omega hierarchical across format and trait.

label	A	B	C	D
Extraversion (5 descriptors)	0.35	0.65	0.78	0.87
Agreeableness (5 descriptors)	0.40	0.35	0.77	0.33
Conscientiousness (10 descriptors)	0.79	0.61	0.47	0.72
Neuroticism (4 descriptors)	0.52	0.84	0.47	0.41
Openness (7 descriptors)	0.42	0.58	0.59	0.53

4.4 Omega

We extract the estimated confidence intervals. (Note that these estimates are unreliable in small samples. The estimates extracted based on pilot data are not expected to reflect estimates provided in the final analyses.) The final summary of results is presented in Table S12 and Figure S17.

```
format_omega = format_data %>%
  mutate(omega_h = map_dbl(omega, "omega_h"))
```

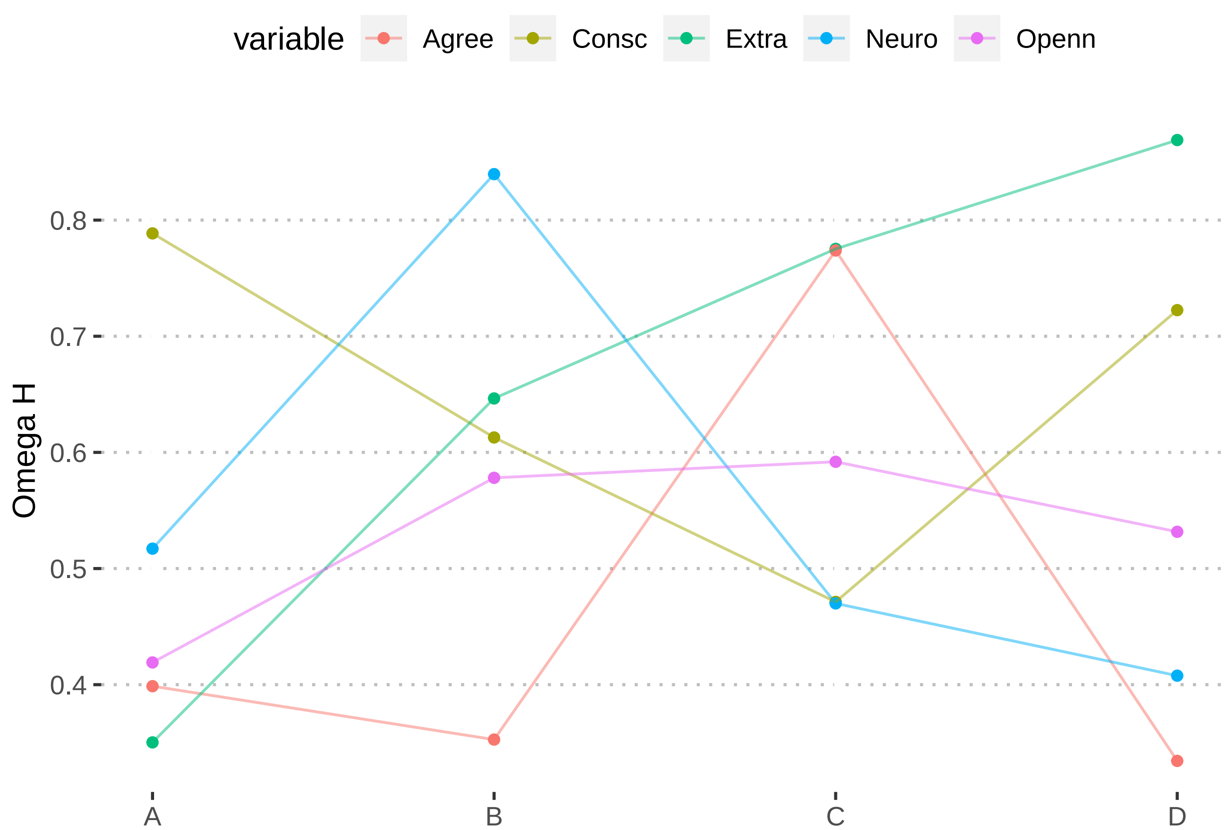


Figure S19: Estimates of omega hierarchical across format and trait.

5 Does the test-retest reliability of personality items change as a function of item wording?

We also plan to evaluate test-retest reliability within formats (within session and over two weeks); we expect slightly higher test-retest reliability for item wording formats that are more specific – formats #3 and #4 above vs the use of adjectives alone. In other words, we expect equal or lower retest reliability for the adjectives than for longer phrases. We will also consider the effect of performance on the word recall task on retest reliability.

The data structure needed for these analyses is in wide-format. That is, we require one column for each time point. In addition, we hope to examine reliability *within* format, which requires selecting only the response options which match the original, Block 1, assessment.

```
items_df = items_df %>%
  mutate(condition = tolower(condition)) %>%
  mutate(condition = factor(condition,
                            levels = c("a", "b", "c", "d"),
                            labels = c("Adjective\nOnly", "Am\nAdjective",
                                       "Tend to be\nAdjective",
                                       "Am someone\nwho tends to be\nAdjective")))

items_matchb1 = items_df %>%
  mutate(across(c(format, condition), as.character)) %>%
  filter(format == condition) %>%
  mutate(block = paste0("block_", block)) %>%
  select(-timing, -seconds_log, -i) %>%
  spread(block, response)
```

We standardize responses within each block – this allows us to use a regression framework yet interpret the slopes as correlations.

```
items_matchb1 = items_matchb1 %>%
  mutate(across(
    starts_with("block"), ~(.-mean(., na.rm=T))/sd(., na.rm = T)
  ))
```

We also standardize the memory scores for ease of interpretation.

```
items_matchb1 = items_matchb1 %>%
  mutate(across(
    ends_with("memory"), ~(.-mean(., na.rm=T))/sd(., na.rm = T)
  ))
```

5.1 Test-retest reliability (all items pooled)

To estimate the reliability coefficients, we use a multilevel model, predicting the latter block from the earlier one. These models nest responses within participant, allowing us to estimate standard errors which account for the dependency of scores. Results are shown in Table S14.

```
tr_mod1_b1b2 = glmmTMB(block_2 ~ block_1 + (1 |proid), data = items_matchb1)
tr_mod1_b1b3 = glmmTMB(block_3 ~ block_1 + (1 |proid), data = items_matchb1)
```

Table S14: Test-retest estimates from multilevel models

Assessments	Slope coefficient
Block 1 - Block 2	0.78 [0.70, 0.85]
Block 1 - Block 3	0.64 [0.58, 0.70]

Table S15: Effect of memory on test-retest

Term	Interpretation	Block 1 - Block 2	Block 1 - Block 3
block_1	Test-retest at average memory	0.77 [0.69, 0.85]	0.64 [0.58, 0.70]
block_1:memory	Change in test-retest by increase in memory	-0.05 [-0.14, 0.04]	0.01 [-0.06, 0.08]
memory	Effect of memory on response	0.02 [-0.09, 0.12]	0.00 [-0.10, 0.09]

5.2 Test-retest reliability (all items pooled, moderated by memory)

Here we fit models moderated by memory – that it, perhaps the test-retest coefficient is affected by the memory of the participant. Results are shown in Table S15

```
tr_mod2_b1b2 = glmmTMB(block_2 ~ block_1*delayed_memory +
                         (1 |proid),
                       data = items_matchb1)
tr_mod2_b1b3 = glmmTMB(block_3 ~ block_1*very_delayed_memory +
                         (1 |proid),
                       data = items_matchb1)
```

We also extract the simple slopes estimates of these models, which allow us to more explicitly identify and compare the test-retest correlations.

5.2.1 Block 1/Block 2

```
mem_list = list(delayed_memory = c(-1,0,1))

emtrends(tr_mod2_b1b2,
         pairwise~delayed_memory,
         var = "block_1",
         at = mem_list)
```

```
## $emtrends
##  delayed_memory block_1.trend     SE  df lower.CL upper.CL
##              -1         0.814 0.0557 246    0.705    0.924
##               0         0.766 0.0408 246    0.685    0.846
##               1         0.717 0.0677 246    0.583    0.850
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate     SE  df t.ratio p.value
##  (-1) - 0   0.0488 0.0467 246   1.045  0.5491
```

```
##   (-1) - 1   0.0976 0.0933 246   1.045  0.5491
##   0 - 1      0.0488 0.0467 246   1.045  0.5491
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

5.2.2 Block 1/Block 3

This chunk is turned off due to low coverage. Be sure to turn on with real data.

```
mem_list = list(very_delayed_memory = c(-1,0,1))

emtrends(tr_mod2_b1b3,
         pairwise~very_delayed_memory,
         var = "block_1",
         at = mem_list)
```

```
## $emtrends
##  very_delayed_memory block_1.trend     SE  df lower.CL upper.CL
##                   -1         0.630 0.0441 676    0.544    0.717
##                    0         0.642 0.0310 676    0.581    0.703
##                    1         0.654 0.0500 676    0.556    0.752
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate     SE  df t.ratio p.value
##  (-1) - 0  -0.0117 0.0355 676  -0.328  0.9423
##  (-1) - 1  -0.0233 0.0710 676  -0.328  0.9423
##  0 - 1     -0.0117 0.0355 676  -0.328  0.9423
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

5.3 Test-retest reliability (all items pooled, by format)

We fit these same models, except now we moderate by format, to determine whether the test-retest reliability differs as a function of item wording.

```
tr_mod3_b1b2 = glmmTMB(block_2 ~ block_1*condition + (1 |proid),
                    data = items_matchb1)
tr_mod3_b1b3 = glmmTMB(block_3 ~ block_1*condition + (1 |proid),
                    data = items_matchb1)

aov(tr_mod3_b1b2)
```

```
## Call:
##    aov(formula = tr_mod3_b1b2)
##
## Terms:
##                   block_1 condition    proid block_1:condition Residuals
## Sum of Squares  150.16283   1.31495 22.36006           2.49860  74.66356
## Deg. of Freedom         1         3       31                 3       213
```

```
##
## Residual standard error: 0.5920584
## 3 out of 42 effects not estimable
## Estimated effects may be unbalanced
## 833 observations deleted due to missingness
```

```
aov(tr_mod3_b1b3)
```

```
## Call:
##    aov(formula = tr_mod3_b1b3)
##
## Terms:
##                  block_1 condition    proid block_1:condition Residuals
## Sum of Squares  264.8656   16.0925  19.2571            1.8381  378.9467
## Deg. of Freedom        1         3       18                 3       656
##
## Residual standard error: 0.7600412
## 3 out of 29 effects not estimable
## Estimated effects may be unbalanced
## 403 observations deleted due to missingness
```

We also extract the simple slopes estimates of these models, which allow us to more explicitly identify and compare the test-retest correlations.

5.3.1 Block 1/Block 2

```
emtrends(tr_mod3_b1b2, pairwise ~ condition, var = "block_1")
```

```
## $emtrends
##   condition                             block_1.trend     SE  df lower.CL
##  Adjective\nOnly                                0.611 0.0730 242    0.468
##  Am\nAdjective                                  0.820 0.0694 242    0.684
##  Am someone\nwho tends to be\nAdjective         0.888 0.0975 242    0.696
##  Tend to be\nAdjective                          0.836 0.0843 242    0.670
##  upper.CL
##     0.755
##     0.957
##     1.080
##     1.002
##
## Confidence level used: 0.95
##
## $contrasts
##   contrast                                                    estimate    SE
##  Adjective\nOnly - Am\nAdjective                               -0.2088 0.101
##  Adjective\nOnly - Am someone\nwho tends to be\nAdjective      -0.2765 0.122
##  Adjective\nOnly - Tend to be\nAdjective                       -0.2243 0.112
##  Am\nAdjective - Am someone\nwho tends to be\nAdjective        -0.0678 0.120
##  Am\nAdjective - Tend to be\nAdjective                         -0.0155 0.109
##  Am someone\nwho tends to be\nAdjective - Tend to be\nAdjective   0.0523 0.131
##   df t.ratio p.value
```

```
## 242 -2.073 0.1648
## 242 -2.273 0.1072
## 242 -2.010 0.1870
## 242 -0.563 0.9429
## 242 -0.143 0.9990
## 242  0.398 0.9786
##
## P value adjustment: tukey method for comparing a family of 4 estimates
```

5.3.2 Block 1/Block 3

```
emtrends(tr_mod3_b1b3, pairwise ~ condition, var = "block_1")
```

```
## $emtrends
##  condition                             block_1.trend     SE  df lower.CL
##  Adjective\nOnly                               0.615 0.0571 672    0.503
##  Am\nAdjective                                 0.607 0.0528 672    0.503
##  Am someone\nwho tends to be\nAdjective        0.644 0.0740 672    0.499
##  Tend to be\nAdjective                         0.715 0.0660 672    0.585
##  upper.CL
##     0.727
##     0.710
##     0.790
##     0.845
##
## Confidence level used: 0.95
##
## $contrasts
##   contrast                                                    estimate     SE
##  Adjective\nOnly - Am\nAdjective                                0.0086 0.0781
##  Adjective\nOnly - Am someone\nwho tends to be\nAdjective      -0.0292 0.0936
##  Adjective\nOnly - Tend to be\nAdjective                       -0.0998 0.0880
##  Am\nAdjective - Am someone\nwho tends to be\nAdjective        -0.0378 0.0909
##  Am\nAdjective - Tend to be\nAdjective                         -0.1084 0.0841
##  Am someone\nwho tends to be\nAdjective - Tend to be\nAdjective  -0.0706 0.0990
##   df t.ratio p.value
##  672   0.110  0.9995
##  672  -0.312  0.9895
##  672  -1.134  0.6683
##  672  -0.416  0.9757
##  672  -1.289  0.5705
##  672  -0.713  0.8920
##
## P value adjustment: tukey method for comparing a family of 4 estimates
```

5.4 Test-retest reliability (items separated, by format)

To assess test-retest reliability for each item, we can rely on more simple correlation analyses, as each participant only contributed one response to each item in each block. We first not the sample size coverage for these comparisons:

```r
items_matchb1 %>%
  group_by(item, condition) %>%
  count() %>%
  ungroup() %>%
  full_join(expand_grid(item = unique(items_matchb1$item),
                        condition = unique(items_matchb1$condition))) %>%
  mutate(n = ifelse(is.na(n), 0, n)) %>%
  summarise(
    min = min(n),
     max = max(n),
    mean = mean(n),
    median = median(n)
  )
```

```
## # A tibble: 1 x 4
##     min   max  mean median
##   <int> <int> <dbl>  <dbl>
## 1     7    11  8.75    8.5
```

```r
items_cors = items_matchb1 %>%
  select(item, condition, contains("block")) %>%
  group_by(item, condition) %>%
  nest() %>%
   mutate(cors = map(data, psych::corr.test, use = "pairwise"),
         cors = map(cors, print, short = F),
         cors = map(cors, ~.x %>% mutate(comp = rownames(.)))) %>%
  select(item, condition, cors) %>%
  unnest(cols = c(cors))
```

The test-retest correlations of each item-format combination are presented in Table S16. We also visualize these correlations in Figure S20,

Table S16: Test-retest correlations for each item and condition. Preregistration note: given the low sample size for the pilot data, we are missing observations for many of these comparisons. Correlations which could not be computed are blank in this table, but we expect them to be reported in the final manuscript.

		Adjective Only		Am Adjective		Tend to be		Am someone who tends to be	
Item	Reverse scored?	5 min	2 weeks	5 min	2 weeks	5 min	2 weeks	5 min	2 weeks
active	N	0.87	-0.32	0.93	0.84*	1.00*	0.75		-0.50
adventurous	N	0.50	0.00		0.00		0.17	0.97	-0.87
broadminded	N		0.29	0.50	-0.44	0.87	0.82*	0.87	1.00*
calm	N	0.58	0.63		0.71		0.40		-0.19
caring	N	-0.50	0.79		-0.54	1.00*	0.34		
cautious	N		0.59		0.48		-0.42		0.91
creative	N	0.87	0.54	1.00*	0.64	0.50	0.71		
curious	N		0.66	0.30	0.00		0.55		0.87
friendly	N		0.50		0.42		-0.54	1.00*	
hardworking	N		0.25		0.88*		0.50		
helpful	N	0.00	0.00	1.00*	0.09		0.25		
imaginative	N	0.90	0.77		0.24	0.98	0.87*		
intelligent	N		0.59		0.50		0.70		0.50
lively	N		0.97*		0.61		0.69		
organized	N	0.97*	0.70		0.77*		0.98*		
outgoing	N		0.72	0.77	0.80*		0.95*		
responsible	N		0.71		0.42		0.48		-0.50
selfdisciplined	N		0.00		0.71	1.00*	0.96*		1.00*
softhearted	N		0.94*		0.31		0.49		0.98
sophisticated	N	1.00*	-0.40	0.96*	0.87*		0.62		0.97
sympathetic	N		0.00	0.50	0.79*	0.50		0.43	1.00*
talkative	N		0.26		0.92*		0.90*		-0.19
thorough	N		0.41	0.50	-0.09	0.50	0.81		-0.50
thrifty	N	0.93	0.46	0.91	0.75		0.90*		0.00
warm	N		0.61		-0.42		0.81		-0.50
careless	Y	0.87	0.05	0.69	0.65		0.16		
impulsive	Y		0.72		0.57	0.96	0.87*		0.62
moody	Y	0.79	0.96*		0.76		0.10		0.84
nervous	Y		0.22	-1.00*	0.48		-0.58		-0.87
reckless	Y	0.87	0.59		0.64	-0.18	-0.08		-0.87
worrying	Y	0.00	0.84*		0.61		0.80		

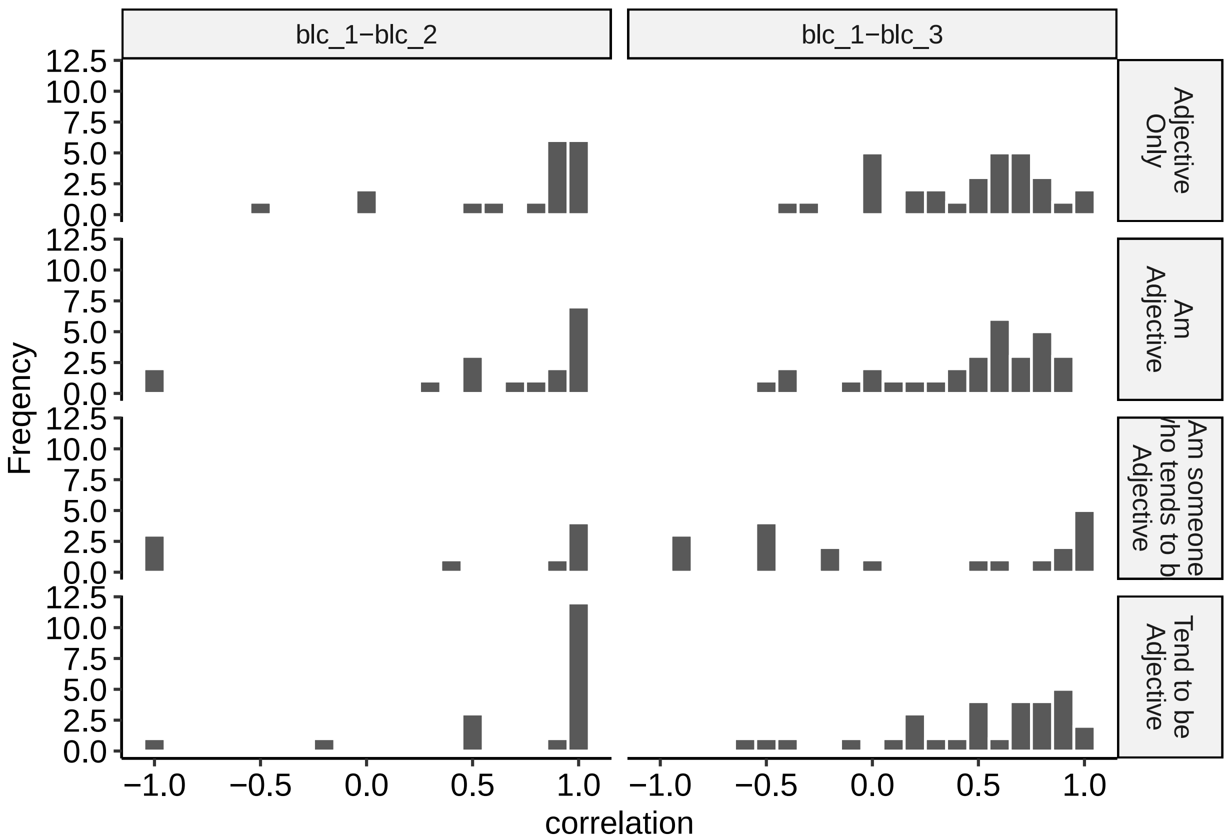


Figure S20: Test-retest correlations of specific items across word format.

Table S17: Pairwise comparisons of timing (log-seconds) across format

contrast	estimate	SE	df	t.ratio	p.value
Adjective Only - Am Adjective	-0.29	0.05	2163	-5.41	0.000
Adjective Only - Tend to be Adjective	-0.27	0.06	2163	-4.83	0.000
Adjective Only - Am someone who tends to be Adjective	-0.35	0.06	2163	-6.37	0.000
Am Adjective - Tend to be Adjective	0.02	0.06	2163	0.38	0.703
Am Adjective - Am someone who tends to be Adjective	-0.06	0.06	2163	-1.15	0.500
Tend to be Adjective - Am someone who tends to be Adjective	-0.09	0.06	2163	-1.52	0.385

6 How does format affect timing of responses?

6.1 Effect of format on timing (Blocks 1 and 2 data)

We used a multilevel model, nesting log-seconds within participant to account for dependence. Our primary predictor was format. Here, we use only Block 1 data. Results are depicted in Figure S21. The full distribution of timing (in log-seconds) is shown in Figure S22. Tests of pairwise comparisons are shown in Table S17.

```
item_block12 = filter(items_df, block %in% c("1", "2"))

mod.format_b1 = glmmTMB(seconds_log~format + (1|block) + (1|proid),
                        data = item_block12)
tidy(aov(mod.format_b1))
```

```
## # A tibble: 4 x 6
##   term         df  sumsq meansq statistic   p.value
##   <chr>     <dbl>  <dbl>  <dbl>     <dbl>     <dbl>
## 1 format        3   26.8  8.94       14.0  4.60e- 9
## 2 block         1   49.8 49.8        78.2  1.91e-18
## 3 proid        34  311.   9.16       14.4  9.37e-73
## 4 Residuals  2131 1357.   0.637      NA   NA
```

6.1.1 One model for each adjective

We can also repeat this analysis separately for each trait. Results are shown in Table S18.

```
mod_by_item_b1 = item_block12 %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~lm(seconds_log~format, data = .))) %>%
  mutate(aov = map(mod, anova)) %>%
  ungroup()
```

6.1.2 Pairwise t-tests for significant ANOVAs

Here we identify the specific items with significant differences.

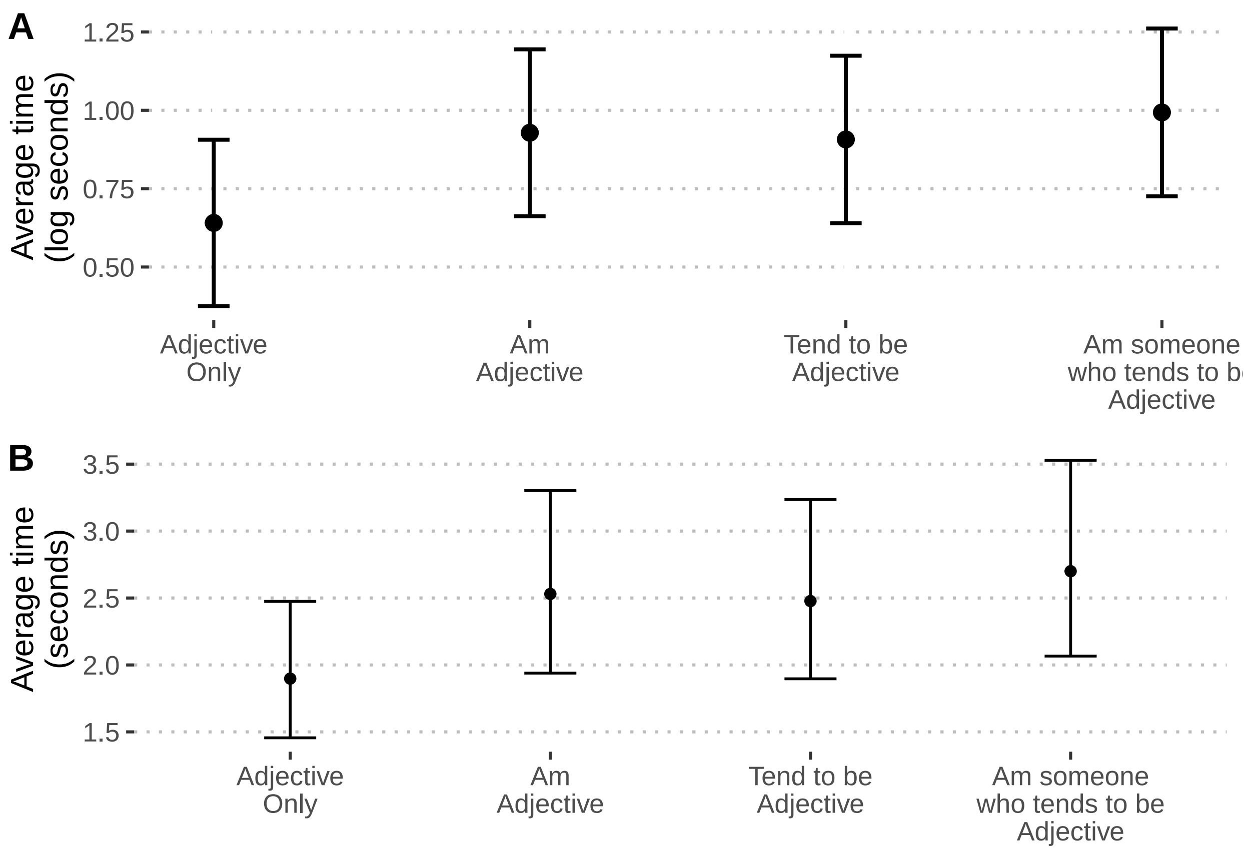


Figure S21: Predictions by condition, using only Block 1 data. Figure A shows log seconds, Figure B shows raw seconds.

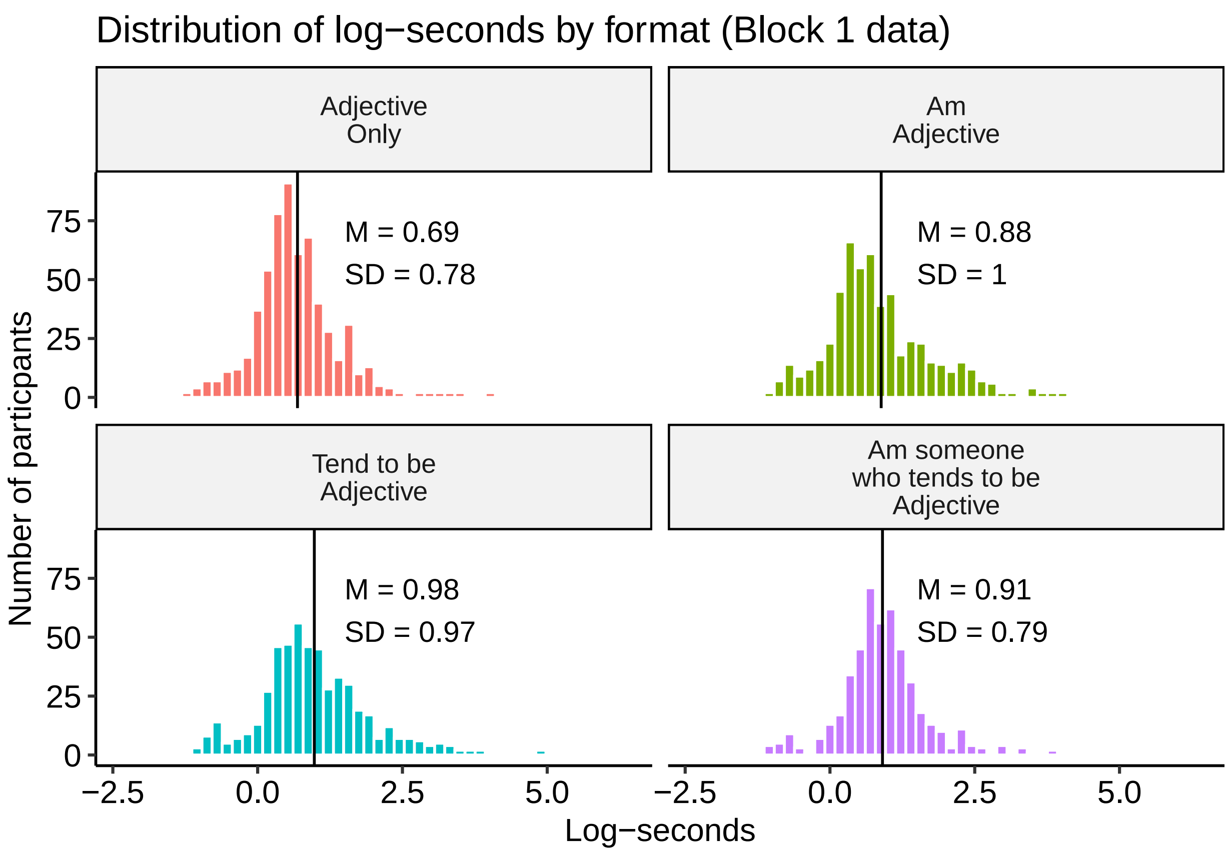


Figure S22: Distribution of time by category, block 1 data only

Table S18: Format effects on log-seconds by item (block 1 data only)

Item	Reverse Scored?	SS	MS	df	F	raw	adj
active	N	1.82	0.61	3	0.81	.493	> .999
adventurous	N	4.47	1.49	3	2.35	.080	> .999
broadminded	N	1.17	0.39	3	0.56	.644	> .999
calm	N	5.06	1.69	3	2.02	.119	> .999
caring	N	5.00	1.67	3	3.01	.036	> .999
cautious	N	3.05	1.02	3	0.93	.431	> .999
creative	N	1.97	0.66	3	0.70	.555	> .999
curious	N	2.47	0.82	3	1.40	.252	> .999
friendly	N	0.41	0.14	3	0.18	.909	> .999
hardworking	N	5.53	1.84	3	2.74	.050	> .999
helpful	N	2.50	0.83	3	3.16	.030	.944
imaginative	N	4.68	1.56	3	1.98	.126	> .999
intelligent	N	3.20	1.07	3	1.02	.389	> .999
lively	N	2.25	0.75	3	0.70	.558	> .999
organized	N	2.16	0.72	3	1.02	.392	> .999
outgoing	N	3.02	1.01	3	1.32	.275	> .999
responsible	N	2.67	0.89	3	1.57	.205	> .999
selfdisciplined	N	1.32	0.44	3	0.66	.580	> .999
softhearted	N	2.34	0.78	3	1.08	.364	> .999
sophisticated	N	1.95	0.65	3	0.67	.572	> .999
sympathetic	N	4.19	1.40	3	1.75	.166	> .999
talkative	N	0.21	0.07	3	0.16	.925	> .999
thorough	N	3.62	1.21	3	1.18	.325	> .999
thrifty	N	3.71	1.24	3	1.26	.295	> .999
warm	N	0.81	0.27	3	0.42	.738	> .999
careless	Y	2.03	0.68	3	1.10	.356	> .999
impulsive	Y	3.81	1.27	3	1.12	.346	> .999
moody	Y	1.44	0.48	3	0.91	.439	> .999
nervous	Y	6.47	2.16	3	2.42	.074	> .999
reckless	Y	1.86	0.62	3	0.57	.639	> .999
worrying	Y	0.76	0.25	3	0.45	.717	> .999

Table S19: Differences in log-seconds to Helpful by format (Block 1 data only)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-0.30	0.16	66	-1.83	.324
Adjective Only - Tend to be Adjective	-0.33	0.18	66	-1.88	.324
Adjective Only - Am someone who tends to be Adjective	-0.52	0.17	66	-3.00	.023
Am Adjective - Tend to be Adjective	-0.03	0.18	66	-0.16	.873
Am Adjective - Am someone who tends to be Adjective	-0.22	0.17	66	-1.24	.664
Tend to be Adjective - Am someone who tends to be Adjective	-0.19	0.18	66	-1.01	.664

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

sig_item_b1 = sig_item_b1$item
sig_item_b1
```

```
## [1] "caring"  "helpful"
```

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!

6.1.3 Helpful

Tests of the pairwise comparisons for this item are shown in Table S19 and means are shown in Figure S23.

```
helpful_model_b1 = item_block12 %>%
  filter(item == "helpful") %>%
  lm(seconds_log~format, data = .)

helpful_em_b1 = emmeans(helpful_model_b1, "format")
```

6.1.4 Caring

Tests of the pairwise comparisons for this item are shown in Table S20 and means are shown in Figure S24.

```
caring_model_b1 = item_block12 %>%
  filter(item == "caring") %>%
  lm(seconds_log~format, data = .)
```

6.1.5 Soft-hearted

Tests of the pairwise comparisons for this item are shown in Table S21 and means are shown in Figure S25.

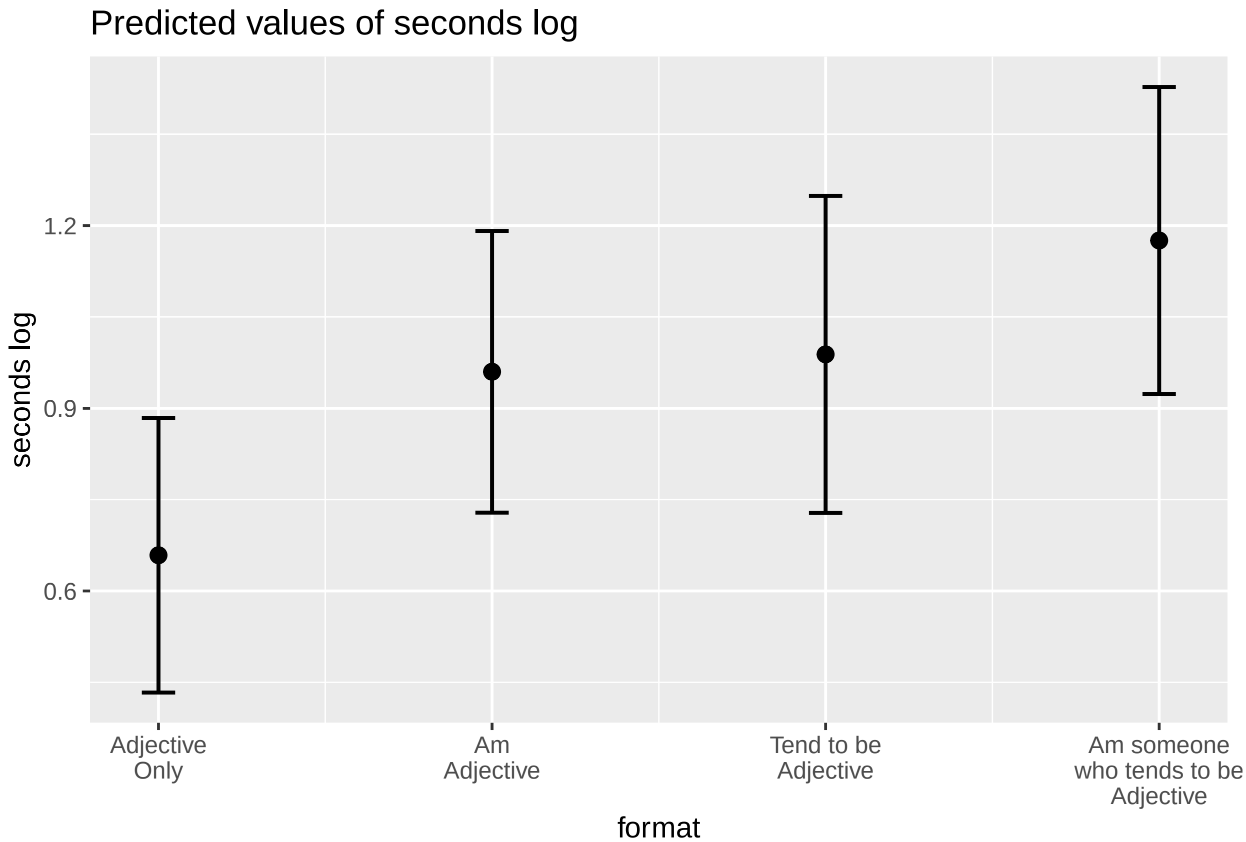


Figure S23: Average log-seconds to “helpful” by format (block 1 data only)

Table S20: Differences in log-seconds to Caring by format (Block 1 data only)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-0.65	0.25	66	-2.64	.062
Adjective Only - Tend to be Adjective	-0.60	0.25	66	-2.41	.094
Adjective Only - Am someone who tends to be Adjective	-0.29	0.25	66	-1.19	.714
Am Adjective - Tend to be Adjective	0.05	0.26	66	0.18	.856
Am Adjective - Am someone who tends to be Adjective	0.36	0.26	66	1.39	.672
Tend to be Adjective - Am someone who tends to be Adjective	0.31	0.26	66	1.19	.714

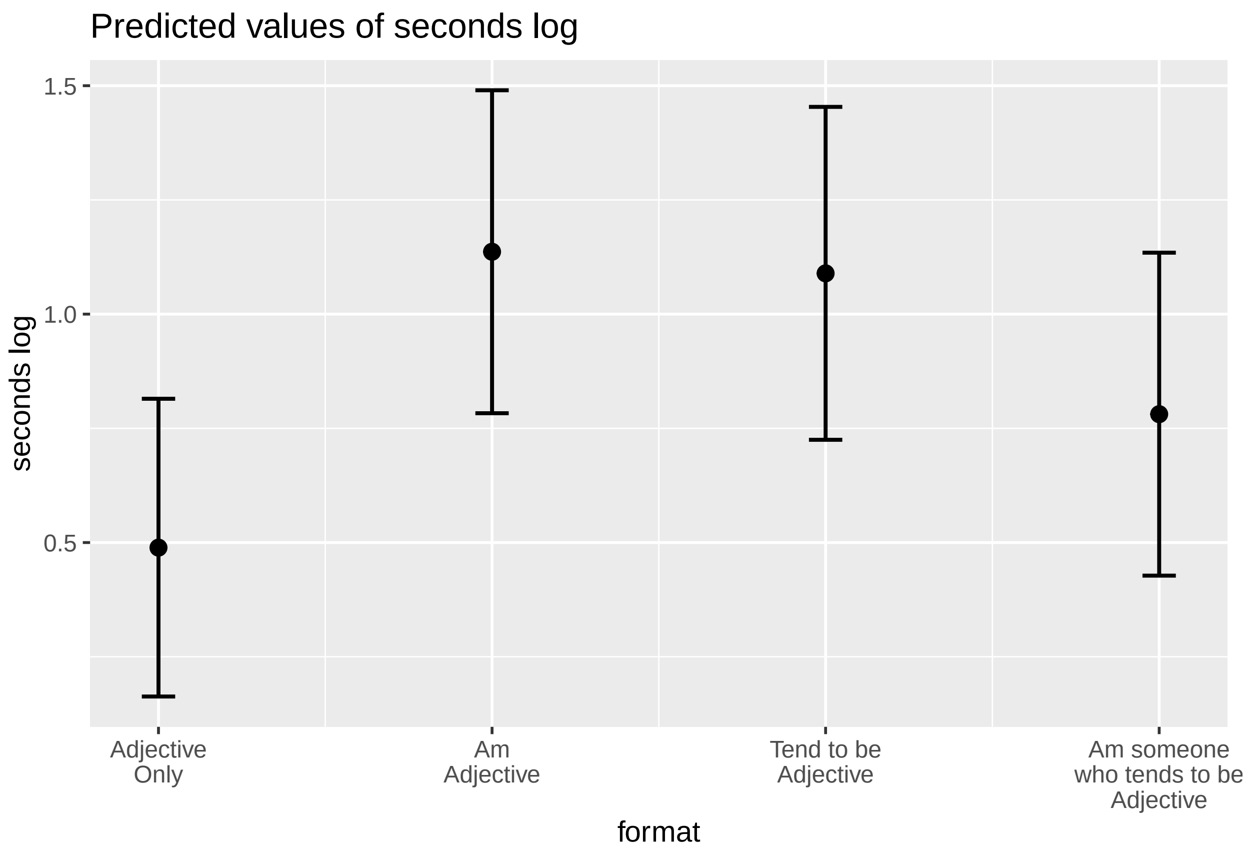


Figure S24: Average log-seconds to "caring" by format (block 1 data only)

Table S21: Differences in log-seconds to Soft-hearted by format (Block 1 data only)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-0.23	0.27	66	-0.86	> .999
Adjective Only - Tend to be Adjective	-0.50	0.28	66	-1.78	.479
Adjective Only - Am someone who tends to be Adjective	-0.16	0.29	66	-0.54	> .999
Am Adjective - Tend to be Adjective	-0.27	0.29	66	-0.92	> .999
Am Adjective - Am someone who tends to be Adjective	0.08	0.30	66	0.26	> .999
Tend to be Adjective - Am someone who tends to be Adjective	0.35	0.31	66	1.13	> .999

```
softhearted_model_b1 = item_block12 %>%
  filter(item == "softhearted") %>%
  lm(seconds_log~format, data = .)
```

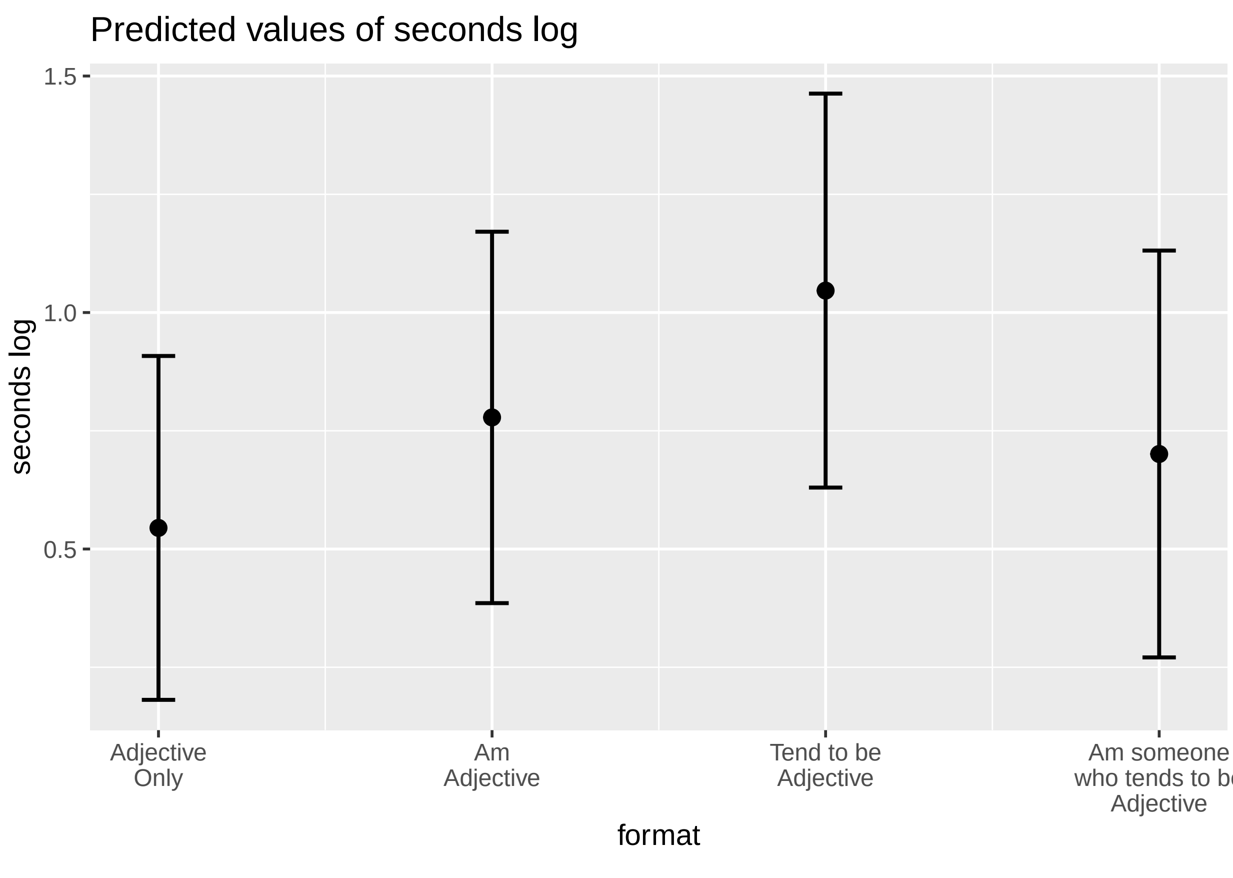


Figure S25: Average log-seconds to "softhearted" by format (block 1 data only)

6.1.6 Calm

Tests of the pairwise comparisons for this item are shown in Table S22 and means are shown in Figure S26.

Table S22: Differences in log-seconds to Calm by format (Block 1 data only)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-0.73	0.30	66	-2.44	.104
Adjective Only - Tend to be Adjective	-0.34	0.30	66	-1.12	.902
Adjective Only - Am someone who tends to be Adjective	-0.41	0.30	66	-1.35	.900
Am Adjective - Tend to be Adjective	0.39	0.32	66	1.22	.902
Am Adjective - Am someone who tends to be Adjective	0.32	0.32	66	1.00	.902
Tend to be Adjective - Am someone who tends to be Adjective	-0.07	0.32	66	-0.22	.902

```
calm_model_b1 = item_block12 %>%
  filter(item == "calm") %>%
  lm(seconds_log~format, data = .)
```

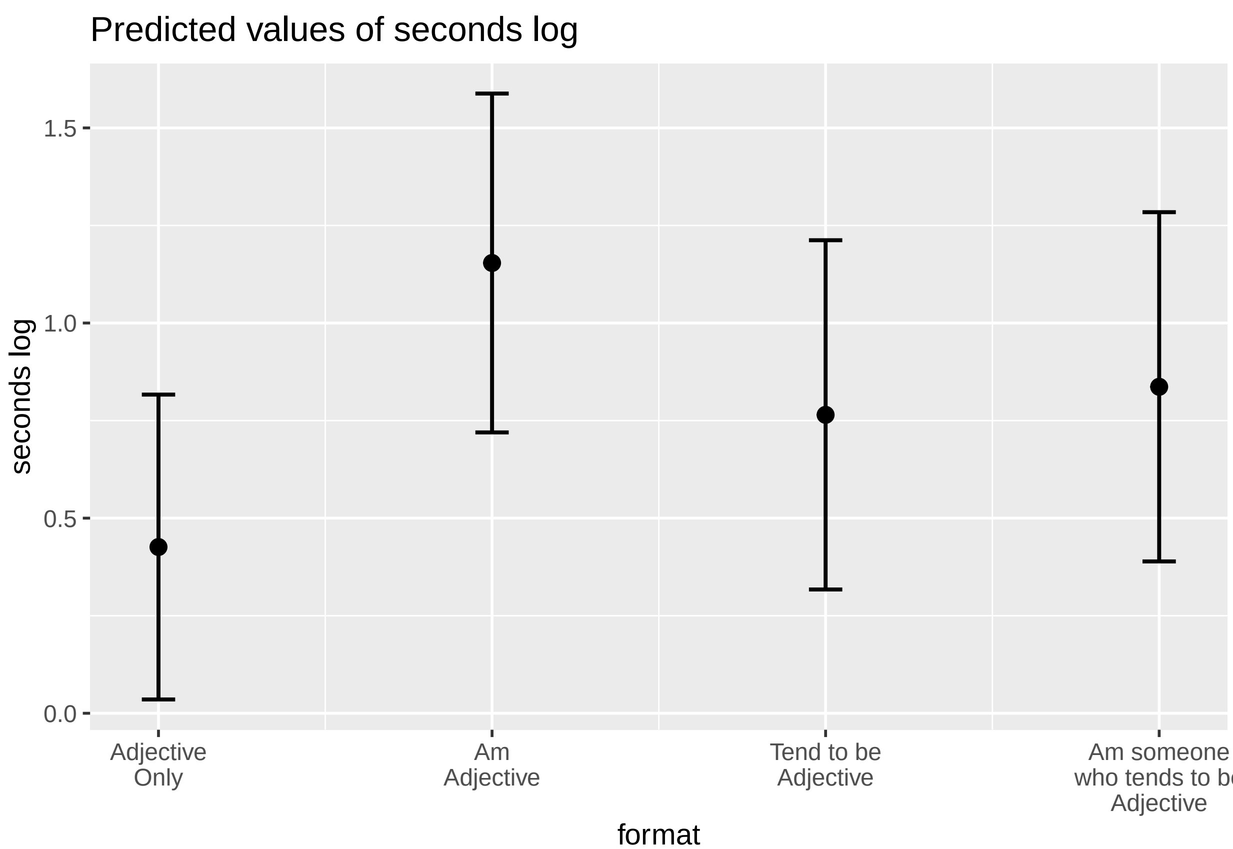


Figure S26: Average log-seconds to “calm” by format (block 1 data only)

6.1.7 Sympathetic

Tests of the pairwise comparisons for this item are shown in Table S23 and means are shown in Figure S27.

Table S23: Differences in log-seconds to Sympathetic by format (Block 1 data only)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-0.28	0.29	66	-0.96	> .999
Adjective Only - Tend to be Adjective	-0.67	0.29	66	-2.28	.155
Adjective Only - Am someone who tends to be Adjective	-0.34	0.31	66	-1.12	> .999
Am Adjective - Tend to be Adjective	-0.40	0.30	66	-1.31	.979
Am Adjective - Am someone who tends to be Adjective	-0.06	0.31	66	-0.20	> .999
Tend to be Adjective - Am someone who tends to be Adjective	0.33	0.32	66	1.05	> .999

```
sympathetic_model_b1 = item_block12 %>%
  filter(item == "sympathetic") %>%
  lm(seconds_log~format, data = .)
```

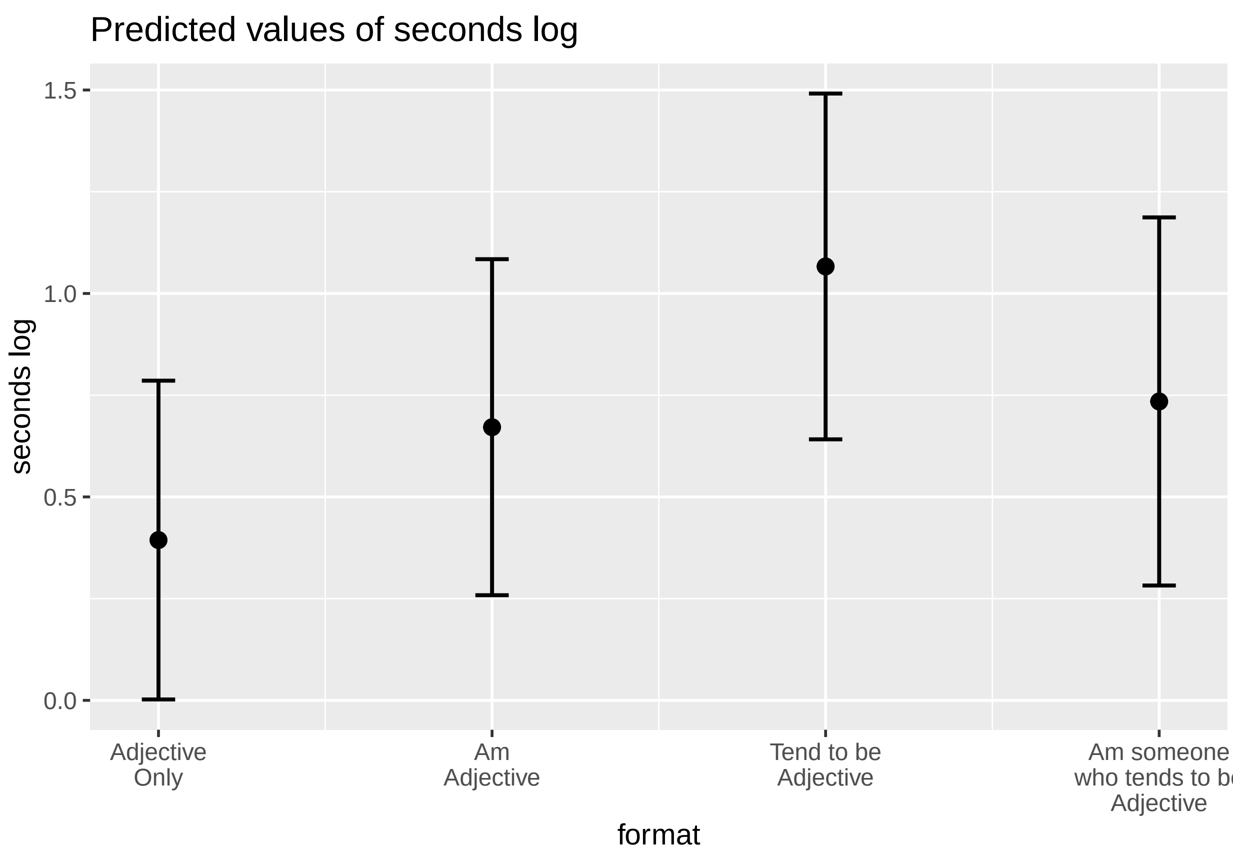


Figure S27: Average log-seconds to “sympathetic” by format (block 1 data only)

6.1.8 Adventurous

Tests of the pairwise comparisons for this item are shown in Table S24 and means are shown in Figure S28.

Table S24: Differences in log-seconds to Adventurous by format (Block 1 data only)

Contrast	Difference in means	SE	df	t	p
Adjective Only - Am Adjective	-0.18	0.26	66	-0.69	.990
Adjective Only - Tend to be Adjective	-0.63	0.26	66	-2.42	.109
Adjective Only - Am someone who tends to be Adjective	-0.48	0.26	66	-1.81	.371
Am Adjective - Tend to be Adjective	-0.45	0.28	66	-1.62	.444
Am Adjective - Am someone who tends to be Adjective	-0.30	0.28	66	-1.06	.880
Tend to be Adjective - Am someone who tends to be Adjective	0.15	0.28	66	0.54	.990

```
adventurous_model_b1 = item_block12 %>%
  filter(item == "adventurous") %>%
  lm(seconds_log~format, data = .)
```

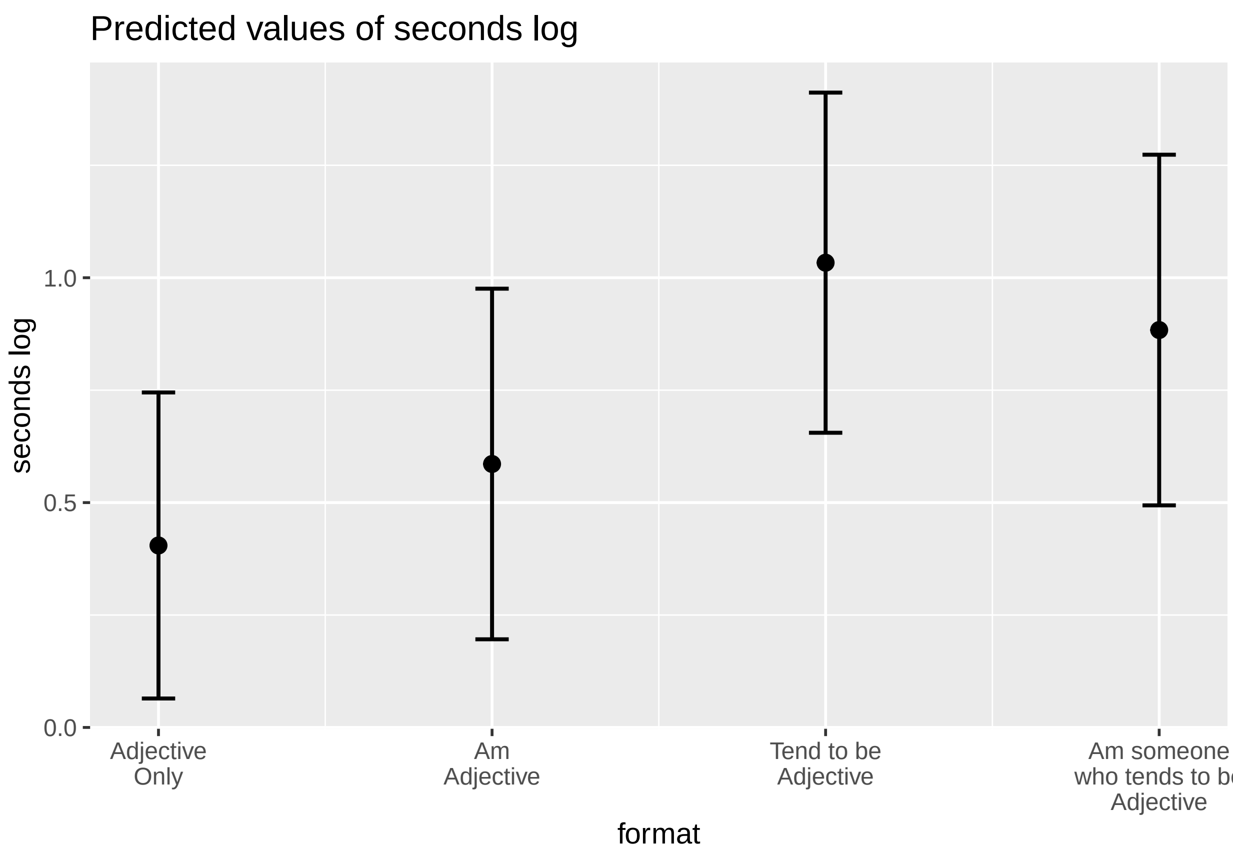


Figure S28: Average log-seconds to “adventurous” by format (block 1 data only)

6.2 Inclusion of “I” (Blocks 1 and 3)

We used a multilevel model, nesting response within participant to account for dependence. Our primary predictors are format and also the presence of the word “I”. Here, we use data from blocks 1 and 3. Results are depicted in Figure S29.

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

```r
mod.format_b3_1 = glmmTMB(seconds_log~format + i + (1|proid),
                  data = items_13)
tidy(aov(mod.format_b3_1)) %>%
  mutate(p.value = papaja::printp(p.value))
```

```
## # A tibble: 4 x 6
##   term         df   sumsq meansq statistic p.value
##   <chr>     <dbl>   <dbl>  <dbl>     <dbl> <chr>
## 1 format        2  22.1   11.1      20.2   "< .001"
## 2 i             1   0.255  0.255     0.464 ".496"
## 3 proid        13  86.7    6.67     12.2   "< .001"
## 4 Residuals   975 535.     0.548    NA     ""
```

```r
mod.format_b3_2 = glmmTMB(seconds_log~format*i + (1|proid),
                  data = items_13)
tidy(aov(mod.format_b3_2)) %>%
  mutate(p.value = papaja::printp(p.value))
```

```
## # A tibble: 5 x 6
##   term         df   sumsq meansq statistic p.value
##   <chr>     <dbl>   <dbl>  <dbl>     <dbl> <chr>
## 1 format        2  22.1   11.1      20.2   "< .001"
## 2 i             1   0.255  0.255     0.464 ".496"
## 3 proid        13  86.7    6.67     12.2   "< .001"
## 4 format:i      2   1.05   0.526     0.959 ".384"
## 5 Residuals   973 534.     0.548    NA     ""
```

6.2.1 One model for each adjective

Additive effects of I (controlling for format) are summarized in Table S25. Tests of the interaction of I with format (for each item) are summarized in Table S26.

```r
mod_by_item_i1 = items_13 %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~glmmTMB(seconds_log~format+i + (1|proid), data = .))) %>%
  mutate(aov = map(mod, aov)) %>%
  ungroup()
```

```r
summary_by_item_i1 = mod_by_item_i1 %>%
  mutate(tidy = map(aov, broom::tidy)) %>%
  select(item, tidy) %>%
  unnest(cols = c(tidy)) %>%
  filter(term == "i") %>%
  mutate(reverse = case_when(
```

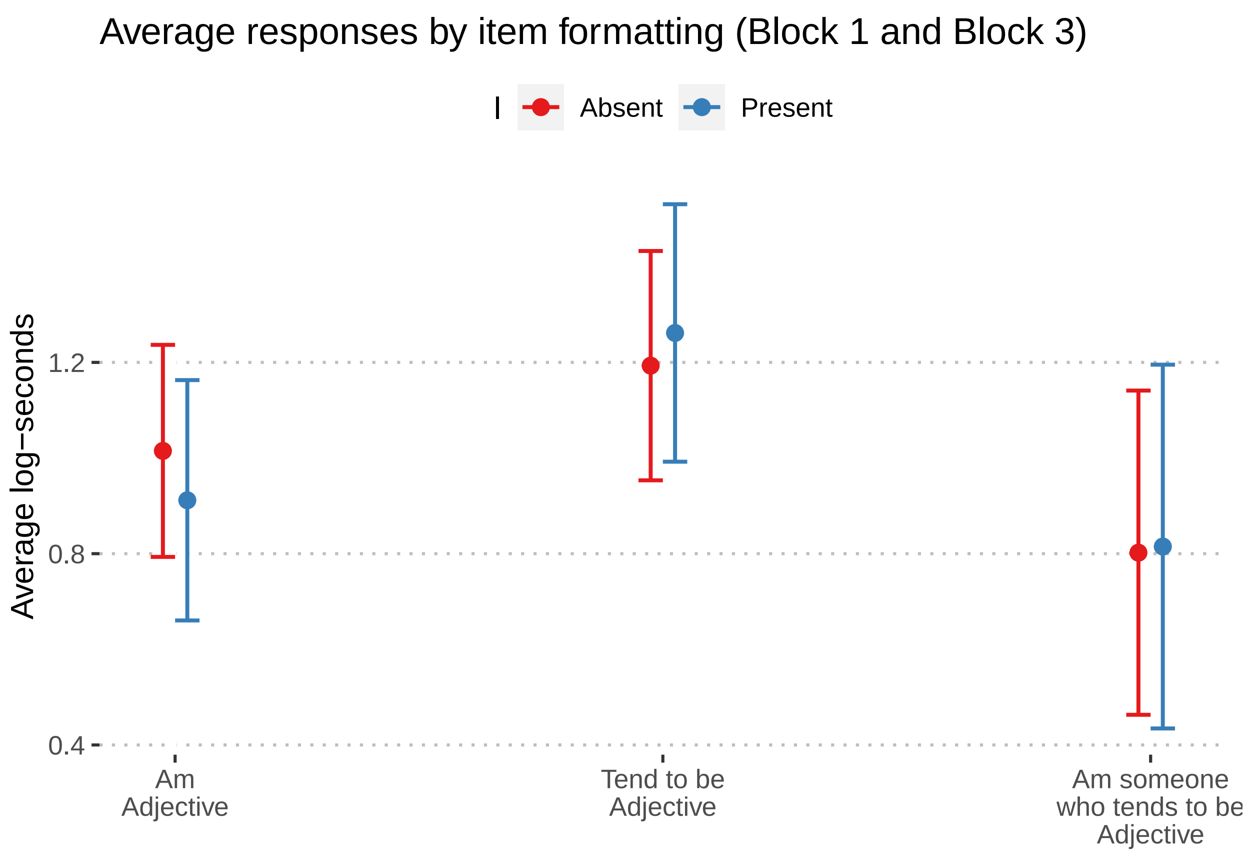


Figure S29: Predicted log-seconds on personality items by condition and I, using Block 1 and Block 3 data.

Table S25: Additive effect of I on timing for each item

item	reverse	sumsq	meansq	df	statistic	p.value	p.adj
active	N	0.04	0.04	1	0.21	.656	> .999
adventurous	N	0.01	0.01	1	0.06	.807	> .999
broadminded	N	0.20	0.20	1	0.44	.516	> .999
calm	N	0.07	0.07	1	0.09	.768	> .999
caring	N	1.37	1.37	1	2.54	.132	> .999
cautious	N	2.81	2.81	1	6.85	.019	.582
creative	N	0.20	0.20	1	0.25	.622	> .999
curious	N	0.77	0.77	1	0.94	.347	> .999
friendly	N	0.03	0.03	1	0.19	.666	> .999
hardworking	N	0.51	0.51	1	0.45	.512	> .999
helpful	N	0.19	0.19	1	1.78	.202	> .999
imaginative	N	0.01	0.01	1	0.01	.916	> .999
intelligent	N	0.15	0.15	1	0.16	.699	> .999
lively	N	0.97	0.97	1	2.54	.132	> .999
organized	N	0.00	0.00	1	0.01	.918	> .999
outgoing	N	0.13	0.13	1	0.15	.702	> .999
responsible	N	0.35	0.35	1	0.72	.410	> .999
selfdisciplined	N	0.36	0.36	1	1.18	.295	> .999
softhearted	N	0.01	0.01	1	0.04	.840	> .999
sophisticated	N	0.09	0.09	1	0.16	.696	> .999
sympathetic	N	0.13	0.13	1	0.25	.626	> .999
talkative	N	0.24	0.24	1	1.06	.320	> .999
thorough	N	0.11	0.11	1	0.16	.693	> .999
thrifty	N	1.52	1.52	1	1.71	.210	> .999
warm	N	1.29	1.29	1	8.36	.011	.347
careless	Y	0.01	0.01	1	0.01	.911	> .999
impulsive	Y	0.78	0.78	1	0.77	.393	> .999
moody	Y	0.02	0.02	1	0.10	.757	> .999
nervous	Y	0.24	0.24	1	0.76	.397	> .999
reckless	Y	0.09	0.09	1	0.32	.579	> .999
worrying	Y	0.47	0.47	1	0.96	.344	> .999

```
    item %in% reverse ~ "Y",
    TRUE ~ "N"
  )) %>%
  mutate(p.adj = p.adjust(p.value, method = "holm"))
```

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

Table S26: Interaction of I with format on timing for each item

item	reverse	sumsq	meansq	df	statistic	p.value	p.adj
active	N	0.31	0.15	2	0.73	.500	> .999
adventurous	N	0.01	0.00	2	0.02	.979	> .999
broadminded	N	0.09	0.05	2	0.09	.914	> .999
calm	N	0.22	0.11	2	0.14	.873	> .999
caring	N	2.77	1.38	2	3.38	.066	> .999
cautious	N	0.56	0.28	2	0.65	.538	> .999
creative	N	0.51	0.26	2	0.29	.752	> .999
curious	N	0.64	0.32	2	0.36	.705	> .999
friendly	N	0.15	0.07	2	0.40	.677	> .999
hardworking	N	1.18	0.59	2	0.49	.626	> .999
helpful	N	0.19	0.09	2	0.85	.451	> .999
imaginative	N	0.23	0.12	2	0.21	.815	> .999
intelligent	N	1.05	0.52	2	0.51	.612	> .999
lively	N	0.28	0.14	2	0.33	.725	> .999
organized	N	0.11	0.05	2	0.55	.591	> .999
outgoing	N	0.04	0.04	1	0.04	.841	> .999
responsible	N	1.12	0.56	2	1.18	.339	> .999
selfdisciplined	N	0.98	0.49	2	1.77	.209	> .999
softhearted	N	1.17	0.59	2	2.18	.152	> .999
sophisticated	N	0.91	0.46	2	0.76	.487	> .999
sympathetic	N	1.08	0.54	2	1.02	.386	> .999
talkative	N	0.03	0.01	2	0.06	.946	> .999
thorough	N	0.07	0.04	2	0.05	.955	> .999
thrifty	N	0.85	0.42	2	0.44	.653	> .999
warm	N	0.64	0.32	2	2.47	.123	> .999
careless	Y	0.35	0.17	2	0.24	.791	> .999
impulsive	Y	1.31	0.66	2	0.62	.554	> .999
moody	Y	0.02	0.01	2	0.06	.943	> .999
nervous	Y	0.62	0.31	2	0.96	.408	> .999
reckless	Y	1.23	0.61	2	2.55	.116	> .999
worrying	Y	0.48	0.24	2	0.44	.651	> .999

7 Power analysis

We conduct power analyses for the research question, "Does item format influence expected response to personality items?" by powering a balanced one-way analysis of variance. This model assumes no individual differences in response, thereby providing a more conservative estimate of the sample size needed.

```
# calculate each individual's average response
means = item_block1 %>%
  group_by(proid, condition) %>%
  summarise(response = mean(response)) %>%
  ungroup()

# calculate mean and variance for each condition
means = means %>%
  group_by(condition) %>%
  summarise(m = mean(response),
            v = var(response),
            n = n())

# calculate ewighted variance
weighted_var = means %>%
  mutate(newv = v*(n-1)) %>%
  select(newv, n) %>%
  colSums()
weighted_var = weighted_var[[1]]/(weighted_var[[2]]-4)

# enter information into power function
power.anova.test(groups = 4,
                 between.var = var(means$m),
                 within.var = weighted_var,
                 power = .9,
                 sig.level = .05)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##          groups = 4
##               n = 135.3274
##     between.var = 0.009118785
##      within.var = 0.2593392
##       sig.level = 0.05
##           power = 0.9
##
## NOTE: n is number in each group
```

This analysis suggests that 136 participants are needed in each condition to achieve 90% power for the differences in means found in the pilot data. To be safe, we plan to recruit 250 participants per condition.

8 R version and packages

All data cleaning and analyses were completed using R version 4.1.3 (2022-03-10) (One Push-Up). Below we list the packages (and versions) used in these analyses.

Package	Version	Authors and contributors
knitr	1.33	Yihui Xie [aut, cre] (<https://orcid.org/0000-0003-0645-5666>), Abhraneel Sarma [ctb], Adam Vogt [ctb], Alastair Andrew [ctb], Alex Zvoleff [ctb], Andre Simon [ctb] (the CSS files under inst/themes/ were derived from the Highlight package http://www.andre-simon.de), Aron Atkins [ctb], Aaron Wolen [ctb], Ashley Manton [ctb], Atsushi Yasumoto [ctb] (<https://orcid.org/0000-0002-8335-495X>), Ben Baumer [ctb], Brian Diggs [ctb], Brian Zhang [ctb], Bulat Yapparov [ctb], Cassio Pereira [ctb], Christophe Dervieux [ctb], David Hall [ctb], David Hugh-Jones [ctb], David Robinson [ctb], Doug Hemken [ctb], Duncan Murdoch [ctb], Elio Campitelli [ctb], Ellis Hughes [ctb], Emily Riederer [ctb], Fabian Hirschmann [ctb], Fitch Simeon [ctb], Forest Fang [ctb], Frank E Harrell Jr [ctb] (the Sweavel package at inst/misc/Sweavel.sty), Garrick Aden-Buie [ctb], Gregoire Detrez [ctb], Hadley Wickham [ctb], Hao Zhu [ctb], Heewon Jeon [ctb], Henrik Bengtsson [ctb], Hiroaki Yutani [ctb], Ian Lyttle [ctb], Hodges Daniel [ctb], Jake Burkhead [ctb], James Manton [ctb], Jared Lander [ctb], Jason Punyon [ctb], Javier Luraschi [ctb], Jeff Arnold [ctb], Jenny Bryan [ctb], Jeremy Ashkenas [ctb, cph] (the CSS file at inst/misc/docco-classic.css), Jeremy Stephens [ctb], Jim Hester [ctb], Joe Cheng [ctb], Johannes Ranke [ctb], John Honaker [ctb], John Muschelli [ctb], Jonathan Keane [ctb], JJ Allaire [ctb], Johan Toloe [ctb], Jonathan Sidi [ctb], Joseph Larmarange [ctb], Julien Barnier [ctb], Kaiyin Zhong [ctb], Kamil Slowikowski [ctb], Karl Forner [ctb], Kevin K. Smith [ctb], Kirill Mueller [ctb], Kohske Takahashi [ctb], Lorenz Walthert [ctb], Lucas Gallindo [ctb], Marius Hofert [ctb], Martin Modrák [ctb], Michael Chirico [ctb], Michael Friendly [ctb], Michal Bojanowski [ctb], Michel Kuhlmann [ctb], Miller Patrick [ctb], Nacho Caballero [ctb], Nick Salkowski [ctb], Niels Richard Hansen [ctb], Noam Ross [ctb], Obada Mahdi [ctb], Pavel N. Krivitsky [ctb] (<https://orcid.org/0000-0002-9101-3362>), Qiang Li [ctb], Ramnath Vaidyanathan [ctb], Richard Cotton [ctb], Robert Krzyzanowski [ctb], Romain Francois [ctb], Ruaridh Williamson [ctb], Scott Kostyshak [ctb], Sebastian Meyer [ctb], Sietse Brouwer [ctb], Simon de Bernard [ctb], Sylvain Rousseau [ctb], Taiyun Wei [ctb], Thibaut Assus [ctb], Thibaut Lamadon [ctb], Thomas Leeper [ctb], Tim Mastny [ctb], Tom Torsney-Weir [ctb], Trevor Davis [ctb], Viktoras Veitas [ctb], Weicheng Zhu [ctb], Wush Wu [ctb], Zachary Foster [ctb]
pwr	1.3-0	Stephane Champely [aut], Claus Ekstrom [ctb], Peter Dalgaard [ctb], Jeffrey Gill [ctb], Stephan Weibelzahl [ctb], Aditya Anandkumar [ctb], Clay Ford [ctb], Robert Volcic [ctb], Helios De Rosario [cre]
ggridges	0.5.3	Claus O. Wilke [aut, cre] (<https://orcid.org/0000-0002-7470-9261>)
emmeans	1.6.3	Russell V. Lenth [aut, cre, cph], Paul Buerkner [ctb], Maxime Herve [ctb], Jonathon Love [ctb], Hannes Riebl [ctb], Henrik Singmann [ctb]
lmerTest	3.1-3	Alexandra Kuznetsova [aut], Per Bruun Brockhoff [aut, ths], Rune Haubo Bojesen Christensen [aut, cre], Sofie Pødenphant Jensen [ctb]

(continued)

Package	Version	Authors and contributors
lme4	1.1-27.1	Douglas Bates [aut] (<https://orcid.org/0000-0001-8316-9503>), Martin Maechler [aut] (<https://orcid.org/0000-0002-8685-9910>), Ben Bolker [aut, cre] (<https://orcid.org/0000-0002-2127-0443>), Steven Walker [aut] (<https://orcid.org/0000-0002-4394-9078>), Rune Haubo Bojesen Christensen [ctb] (<https://orcid.org/0000-0002-4494-3399>), Henrik Singmann [ctb] (<https://orcid.org/0000-0002-4842-3657>), Bin Dai [ctb], Fabian Scheipl [ctb] (<https://orcid.org/0000-0001-8172-3603>), Gabor Grothendieck [ctb], Peter Green [ctb] (<https://orcid.org/0000-0002-0238-9852>), John Fox [ctb], Alexander Bauer [ctb], Pavel N. Krivitsky [ctb, cph] (<https://orcid.org/0000-0002-9101-3362>, shared copyright on simulate.formula)
Matrix	1.3-4	Douglas Bates [aut], Martin Maechler [aut, cre] (<https://orcid.org/0000-0002-8685-9910>), Timothy A. Davis [ctb] (SuiteSparse and 'cs' C libraries, notably CHOLMOD, AMD; collaborators listed in dir(pattern = '^[A-Z]+[.]txt$', full.names=TRUE, system.file('doc', 'SuiteSparse', package='Matrix'))), Jens Oehlschlägel [ctb] (initial nearPD()), Jason Riedy [ctb] (condest() and onenormest() for octave, Copyright: Regents of the University of California), R Core Team [ctb] (base R matrix implementation)
broom.mixed	0.2.7	Ben Bolker [aut, cre] (<https://orcid.org/0000-0002-2127-0443>), David Robinson [aut], Dieter Menne [ctb], Jonah Gabry [ctb], Paul Buerkner [ctb], Christopher Hua [ctb], William Petry [ctb] (<https://orcid.org/0000-0002-5230-5987>), Joshua Wiley [ctb] (<https://orcid.org/0000-0002-0271-6702>), Patrick Kennedy [ctb], Eduard Szöcs [ctb] (<https://orcid.org/0000-0001-5376-1194>, BASF SE), Indrajeet Patil [ctb], Vincent Arel-Bundock [ctb] (<https://orcid.org/0000-0003-2042-7063>)
psych	2.1.6	William Revelle [aut, cre] (<https://orcid.org/0000-0003-4880-9610>)
papaja	0.1.0.9997	Frederik Aust [aut, cre] (<https://orcid.org/0000-0003-4900-788X>), Marius Barth [aut] (<https://orcid.org/0000-0002-3421-6665>), Birk Diedenhofen [ctb], Christoph Stahl [ctb], Joseph V. Casillas [ctb], Rudolf Siegel [ctb]
stringdist	0.9.7	Mark van der Loo [aut, cre] (<https://orcid.org/0000-0002-9807-4686>), Jan van der Laan [ctb], R Core Team [ctb], Nick Logan [ctb], Chris Muir [ctb], Johannes Gruber [ctb]
kableExtra	1.3.4	Hao Zhu [aut, cre] (<https://orcid.org/0000-0002-3386-6076>), Thomas Travison [ctb], Timothy Tsai [ctb], Will Beasley [ctb], Yihui Xie [ctb], GuangChuang Yu [ctb], Stéphane Laurent [ctb], Rob Shepherd [ctb], Yoni Sidi [ctb], Brian Salzer [ctb], George Gui [ctb], Yeliang Fan [ctb], Duncan Murdoch [ctb], Bill Evans [ctb]
ggpubr	0.4.0	Alboukadel Kassambara [aut, cre]
sjPlot	2.8.9	Daniel Lüdecke [aut, cre] (<https://orcid.org/0000-0002-8895-3206>), Alexander Bartel [ctb] (<https://orcid.org/0000-0002-1280-6138>), Carsten Schwemmer [ctb], Chuck Powell [ctb] (<https://orcid.org/0000-0002-3606-2188>), Amir Djalovski [ctb], Johannes Titz [ctb] (<https://orcid.org/0000-0002-1102-5719>)

(continued)

Package	Version	Authors and contributors
broom	0.7.9	David Robinson [aut], Alex Hayes [aut] (<https://orcid.org/0000-0002-4985-5160>), Simon Couch [aut, cre] (<https://orcid.org/0000-0001-5676-5107>), Indrajeet Patil [ctb] (<https://orcid.org/0000-0003-1995-6531>), Derek Chiu [ctb], Matthieu Gomez [ctb], Boris Demeshev [ctb], Dieter Menne [ctb], Benjamin Nutter [ctb], Luke Johnston [ctb], Ben Bolker [ctb], Francois Briatte [ctb], Jeffrey Arnold [ctb], Jonah Gabry [ctb], Luciano Selzer [ctb], Gavin Simpson [ctb], Jens Preussner [ctb], Jay Hesselberth [ctb], Hadley Wickham [ctb], Matthew Lincoln [ctb], Alessandro Gasparini [ctb], Lukasz Komsta [ctb], Frederick Novometsky [ctb], Wilson Freitas [ctb], Michelle Evans [ctb], Jason Cory Brunson [ctb], Simon Jackson [ctb], Ben Whalley [ctb], Karissa Whiting [ctb], Yves Rosseel [ctb], Michael Kuehn [ctb], Jorge Cimentada [ctb], Erle Holgersen [ctb], Karl Dunkle Werner [ctb] (<https://orcid.org/0000-0003-0523-7309>), Ethan Christensen [ctb], Steven Pav [ctb], Paul PJ [ctb], Ben Schneider [ctb], Patrick Kennedy [ctb], Lily Medina [ctb], Brian Fannin [ctb], Jason Muhlenkamp [ctb], Matt Lehman [ctb], Bill Denney [ctb] (<https://orcid.org/0000-0002-5759-428X>), Nic Crane [ctb], Andrew Bates [ctb], Vincent Arel-Bundock [ctb] (<https://orcid.org/0000-0003-2042-7063>), Hideaki Hayashi [ctb], Luis Tobalina [ctb], Annie Wang [ctb], Wei Yang Tham [ctb], Clara Wang [ctb], Abby Smith [ctb] (<https://orcid.org/0000-0002-3207-0375>), Jasper Cooper [ctb] (<https://orcid.org/0000-0002-8639-3188>), E Auden Krauska [ctb] (<https://orcid.org/0000-0002-1466-5850>), Alex Wang [ctb], Malcolm Barrett [ctb] (<https://orcid.org/0000-0003-0299-5825>), Charles Gray [ctb] (<https://orcid.org/0000-0002-9978-011X>), Jared Wilber [ctb], Vilmantas Gegzna [ctb] (<https://orcid.org/0000-0002-9500-5167>), Eduard Szoecs [ctb], Frederik Aust [ctb] (<https://orcid.org/0000-0003-4900-788X>), Angus Moore [ctb], Nick Williams [ctb], Marius Barth [ctb] (<https://orcid.org/0000-0002-3421-6665>), Bruna Wundervald [ctb] (<https://orcid.org/0000-0001-8163-220X>), Joyce Cahoon [ctb] (<https://orcid.org/0000-0001-7217-4702>), Grant McDermott [ctb] (<https://orcid.org/0000-0001-7883-8573>), Kevin Zarca [ctb], Shiro Kuriwaki [ctb] (<https://orcid.org/0000-0002-5687-2647>), Lukas Wallrich [ctb] (<https://orcid.org/0000-0003-2121-5177>), James Martherus [ctb] (<https://orcid.org/0000-0002-8285-3300>), Chuliang Xiao [ctb] (<https://orcid.org/0000-0002-8466-9398>), Joseph Larmarange [ctb], Max Kuhn [ctb], Michal Bojanowski [ctb], Hakon Malmedal [ctb], Clara Wang [ctb], Sergio Oller [ctb], Luke Sonnet [ctb], Jim Hester [ctb], Cory Brunson [ctb], Ben Schneider [ctb], Bernie Gray [ctb] (<https://orcid.org/0000-0001-9190-6032>), Mara Averick [ctb], Aaron Jacobs [ctb], Andreas Bender [ctb], Sven Templer [ctb], Paul-Christian Buerkner [ctb], Matthew Kay [ctb], Erwan Le Pennec [ctb], Johan Junkka [ctb], Hao Zhu [ctb], Benjamin Soltoff [ctb], Zoe Wilkinson Saldana [ctb], Tyler Littlefield [ctb], Charles T. Gray [ctb], Shabbh E. Banks [ctb], Serina Robinson [ctb], Roger Bivand [ctb], Riinu Ots [ctb], Nicholas Williams [ctb], Nina Jakobsen [ctb], Michael Weylandt [ctb], Lisa Lendway [ctb], Karl Hailperin [ctb], Josue Rodriguez [ctb], Jenny Bryan [ctb], Chris Jarvis [ctb], Greg Macfarlane [ctb], Brian Mannakee [ctb], Drew Tyre [ctb], Shreyas Singh [ctb], Laurens Geffert [ctb], Hong Ooi [ctb], Henrik Bengtsson [ctb], Eduard Szocs [ctb], David Hugh-Jones [ctb], Matthieu Stigler [ctb], Hugo Tavares [ctb] (<https://orcid.org/0000-0001-9373-2726>), R. Willem Vervoort [ctb], Brenton M. Wiernik [ctb], Josh Yamamoto [ctb], Jasme Lee [ctb]

(continued)

Package	Version	Authors and contributors
glmmTMB	1.1.2	Arni Magnusson [aut] (<https://orcid.org/0000-0003-2769-6741>), Hans Skaug [aut], Anders Nielsen [aut] (<https://orcid.org/0000-0001-9683-9262>), Casper Berg [aut] (<https://orcid.org/0000-0002-3812-5269>), Kasper Kristensen [aut], Martin Maechler [aut] (<https://orcid.org/0000-0002-8685-9910>), Koen van Bentham [aut], Ben Bolker [aut, cre] (<https://orcid.org/0000-0002-2127-0443>), Nafis Sadat [ctb] (<https://orcid.org/0000-0001-5715-616X>), Daniel Lüdecke [ctb] (<https://orcid.org/0000-0002-8895-3206>), Russ Lenth [ctb], Joseph O'Brien [ctb] (<https://orcid.org/0000-0001-9851-5077>), Charles J. Geyer [ctb], Maeve McGillycuddy [ctb], Mollie Brooks [aut] (<https://orcid.org/0000-0001-6963-8326>)
stringi	1.7.4	Marek Gagolewski [aut, cre, cph] (<https://orcid.org/0000-0003-0637-6028>), Bartek Tartanus [ctb], and others (stringi source code); IBM, Unicode, Inc. and others (ICU4C source code, Unicode Character Database)
janitor	2.1.0	Sam Firke [aut, cre], Bill Denney [ctb], Chris Haid [ctb], Ryan Knight [ctb], Malte Grosser [ctb], Jonathan Zadra [ctb]
forcats	0.5.1	Hadley Wickham [aut, cre], RStudio [cph, fnd]
stringr	1.4.0	Hadley Wickham [aut, cre, cph], RStudio [cph, fnd]
dplyr	1.0.7	Hadley Wickham [aut, cre] (<https://orcid.org/0000-0003-4757-117X>), Romain François [aut] (<https://orcid.org/0000-0002-2444-4226>), Lionel Henry [aut], Kirill Müller [aut] (<https://orcid.org/0000-0002-1416-3412>), RStudio [cph, fnd]
purrr	0.3.4	Lionel Henry [aut, cre], Hadley Wickham [aut], RStudio [cph, fnd]
readr	2.0.1	Hadley Wickham [aut], Jim Hester [aut, cre], Romain Francois [ctb], RStudio [cph, fnd], https://github.com/mandreyel/ [cph] (mio library), Jukka Jylänki [ctb, cph] (grisu3 implementation), Mikkel Jørgensen [ctb, cph] (grisu3 implementation)
tidyr	1.1.3	Hadley Wickham [aut, cre], RStudio [cph]
tibble	3.1.4	Kirill Müller [aut, cre], Hadley Wickham [aut], Romain Francois [ctb], Jennifer Bryan [ctb], RStudio [cph]
ggplot2	3.3.5	Hadley Wickham [aut] (<https://orcid.org/0000-0003-4757-117X>), Winston Chang [aut] (<https://orcid.org/0000-0002-1576-2126>), Lionel Henry [aut], Thomas Lin Pedersen [aut, cre] (<https://orcid.org/0000-0002-5147-4711>), Kohske Takahashi [aut], Claus Wilke [aut] (<https://orcid.org/0000-0002-7470-9261>), Kara Woo [aut] (<https://orcid.org/0000-0002-5125-4188>), Hiroaki Yutani [aut] (<https://orcid.org/0000-0002-3385-7233>), Dewey Dunnington [aut] (<https://orcid.org/0000-0002-9415-4582>), RStudio [cph, fnd]
tidyverse	1.3.1	Hadley Wickham [aut, cre], RStudio [cph, fnd]
here	1.0.1	Kirill Müller [aut, cre] (<https://orcid.org/0000-0002-1416-3412>), Jennifer Bryan [ctb] (<https://orcid.org/0000-0002-6983-2759>)