Supplemental file

Last updated 2021-07-19

Contents

Cleaning

Workspace

```
library(here) # for working with files
library(tidyverse) # for cleaning
library(janitor) # for variable names
library(lme4) # for multilevel modeling
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
```

```
data_path = here("data/Wording_July 13, 2021_20.00.text.csv")

data_labels = read_csv(data_path)

data = read_csv(data_path,
                skip = 3,
                col_names = names(data_labels))
rm(data_labels)
data = clean_names(data)
```

Remove the following columns.

```
data = data %>%
  select(-end_date,
         -ip_address,
         -progress,
         -finished,
         -recorded_date,
         -external_reference,
         -distribution_channel,
         -user_language,
         -starts_with("recipient"),
         -starts_with("location"),
         -starts_with("meta_info"))
```

Recode personality item responses to numeric

We recode the responses to personality items, which we downloaded as text strings.

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

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

Next we write a simple function to recode values.

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

Now we merge this back into the data.

```
data = select(data, -all_of(p_items))
data = full_join(data, personality_items)
```

Drop bots

Based on ID

We removed 5 participants without valid Prolific IDs.

```
data = data %>%
  mutate(proid = str_remove(proid, "Value will be set from panel or URL"),
         proid = str_remove(proid, "Value will be set from panel or UR"),
         proid = str_remove(proid, "TEST")) %>%
  filter(proid != "")
```

We removed 0 participants that do not speak english well or very well.

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.

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

# extract block 1 questions
block1 = data %>%
  select(proid, matches("^[[:alpha:]]+_[abcd]$"))

#rename variables
n = 0
for(i in adjectives){
  n = n+1
  names(block1) = str_replace(names(block1), i, paste0("trait", str_pad(n, 2, pad = "0")))
}


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

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

#   working on this!!!
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 frame
block1$block1_runs = block1_runs
```

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

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

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

block2_runs = numeric(length = nrow(block2))

#   working on this!!!
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 frame
```

```
block2$block2_runs = block2_runs
```

```
#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"
    ))
```

```
#visualize
runs_data %>%
  ggplot(aes(block1_runs, block2_runs)) +
  geom_point(aes(color = remove)) +
  scale_color_manual(values = c("black", "red")) +
  guides(color = "none") +
  labs(
    x = "block 1 runs",
    y = "block 2 runs"
  ) +
  theme_pubr()
```

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

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

rm(runs_data)
```

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 durations. 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.

```
in_average = data %>%
  # 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)) %>%
```

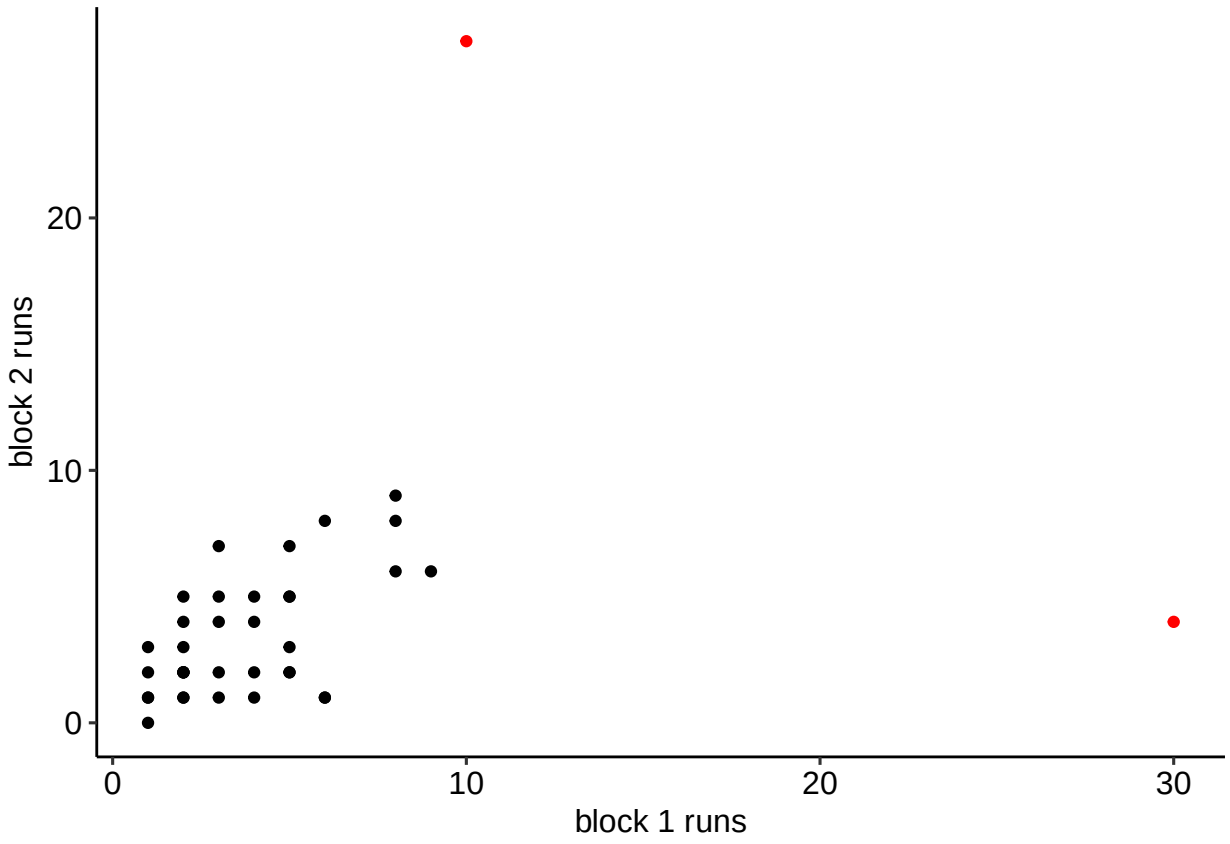

Figure 1: Maximum number of same consecutive responses in personality blocks.

```
  group_by(proid) %>%
  summarise(avg = mean(response)) %>%
  mutate(
    remove = case_when(
      avg >= 4 ~ "Remove",
      TRUE ~ "Keep"))
```

```
in_average %>%
  ggplot(aes(x = avg, fill = remove)) +
  geom_histogram(bins = 20, color = "white") +
  geom_vline(aes(xintercept = 4)) +
  guides(fill = "none") +
  labs(x = "Average response to inattention check items") +
  theme_pubr()
```

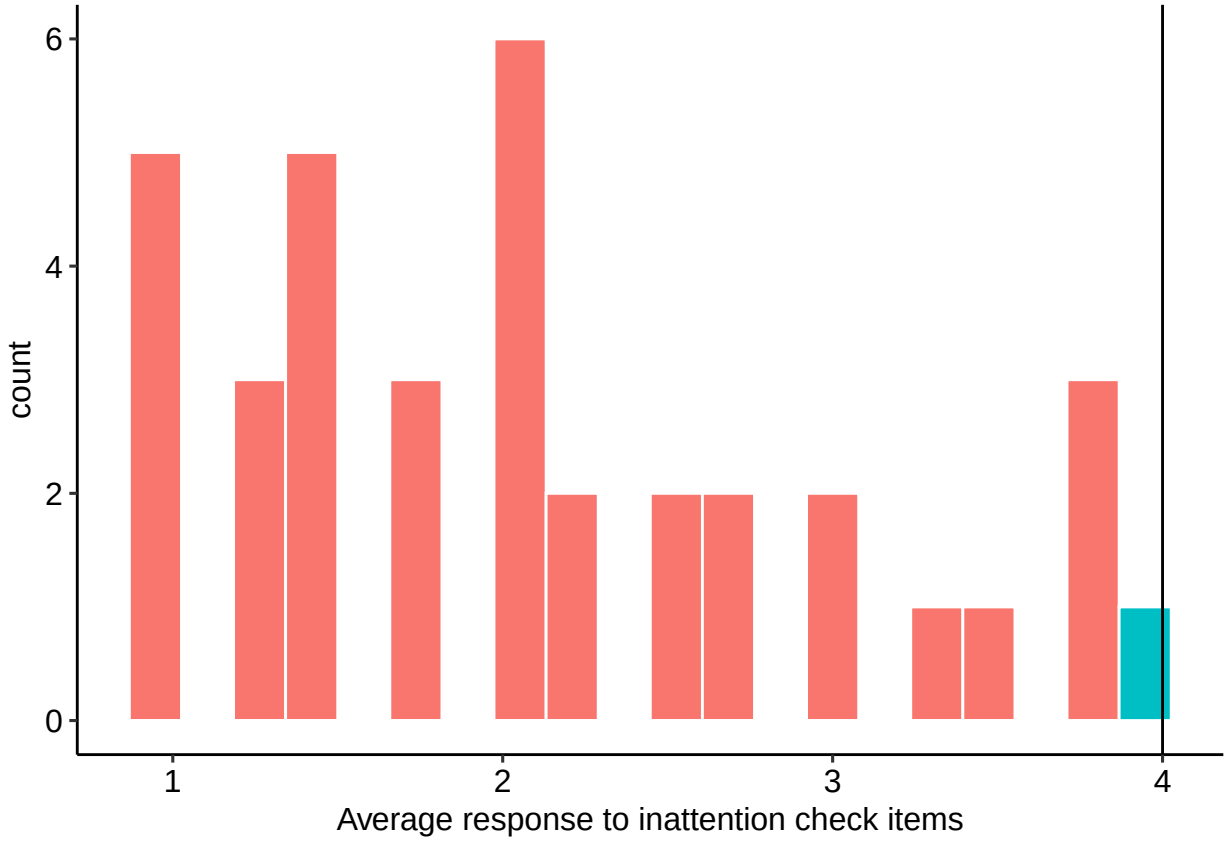

Figure 2: Average response to inattention check items

We remove 1 participants whose responses suggest inattention.

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

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 %>%
  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))

timing_data
```

```
## # A tibble: 2,170 x 3
##    proid                    variable                 timing
##    <chr>                    <chr>                     <dbl>
##  1 5f0227ae81bf2f3a4618c8c7 t_outgoing_a_page_submit   4.47
##  2 60eb3434de863fc43f563b0e t_outgoing_a_page_submit   4.52
##  3 60eb31222e8b2fb8dc904432 t_outgoing_a_page_submit   4.86
##  4 60e49d66c9c4f08ce7a3b789 t_outgoing_a_page_submit   4.16
##  5 60ea7b0e32a76a57b4a34664 t_outgoing_a_page_submit   4.20
##  6 60e950d879a14636c5fc286d t_outgoing_a_page_submit   4.49
##  7 60e781742748ec6401b79f86 t_outgoing_a_page_submit   4.5
##  8 60e9521961c670e718bcc4df t_outgoing_a_page_submit  85.7
##  9 60e777356e13630d745eeb49 t_outgoing_a_page_submit   2.47
## 10 60e99999b101ef725cb0b8a2 t_outgoing_a_page_submit   4.45
## # ... with 2,160 more rows
```

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

```
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 @ref(fig:timing_dist) for a distribution of average response time.

```
timing_data = timing_data %>%
  group_by(proid) %>%
  summarise(m_time = mean(timing)) %>%
```

```
  mutate(remove = case_when(
    m_time < 1 ~ "Remove",
    m_time > 30 ~ "Remove",
    TRUE ~ "Keep"
  ))
```

```
timing_data %>%
  ggplot(aes(x = m_time, fill = remove)) +
  geom_histogram(color = "white") +
  labs(x = "Average response time (seconds)", y = "Number of participants") +
  theme_pubr()
```

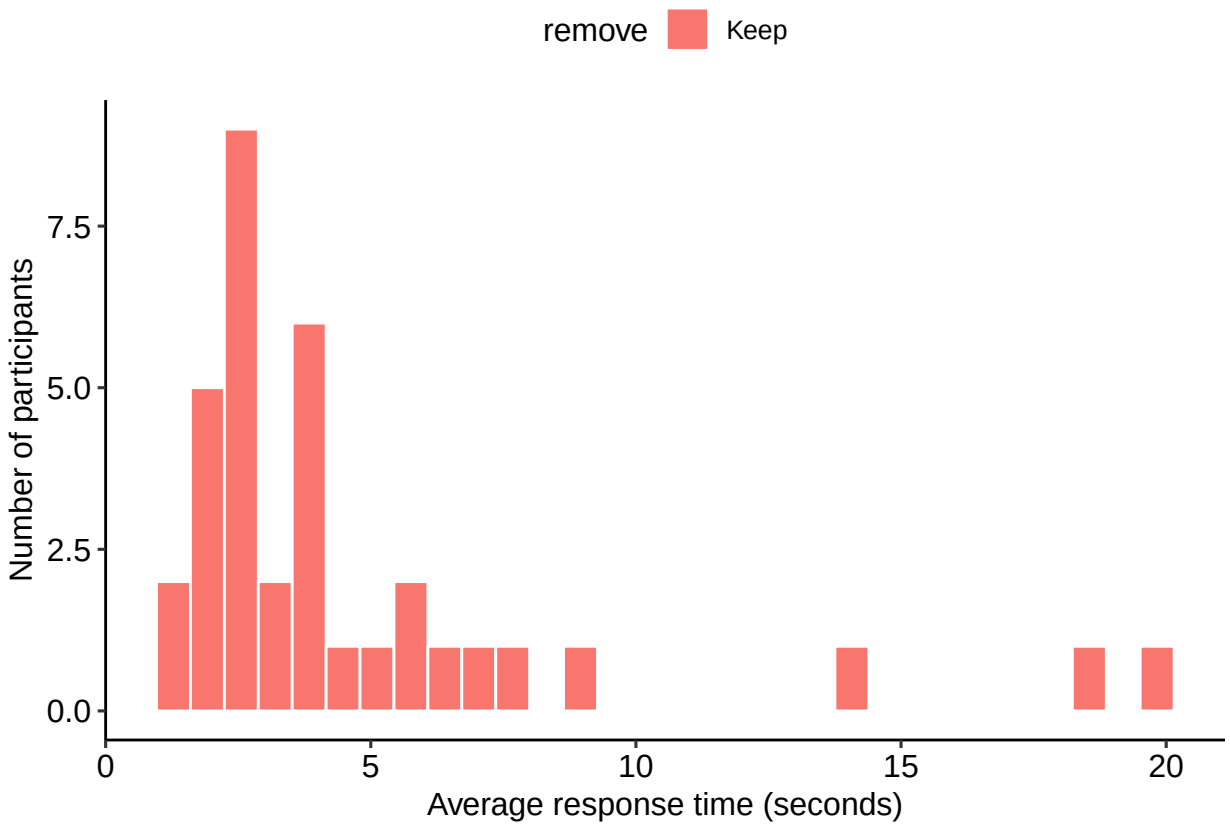

Figure 3: Distribution of average time to respond to personality items.

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

Based on timing, we removed 0 participants.

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

```
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))
```

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 = ","))
```

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...

```
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 resposes, 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 resposnes, 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)
```

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

```r
data %>%
  ggplot(aes(x = memory)) +
  geom_histogram(bins = 11, color = "white") +
  labs(x = "Number of correct responses") +
  scale_x_continuous(breaks = 0:10) +
  theme_pubr()
```

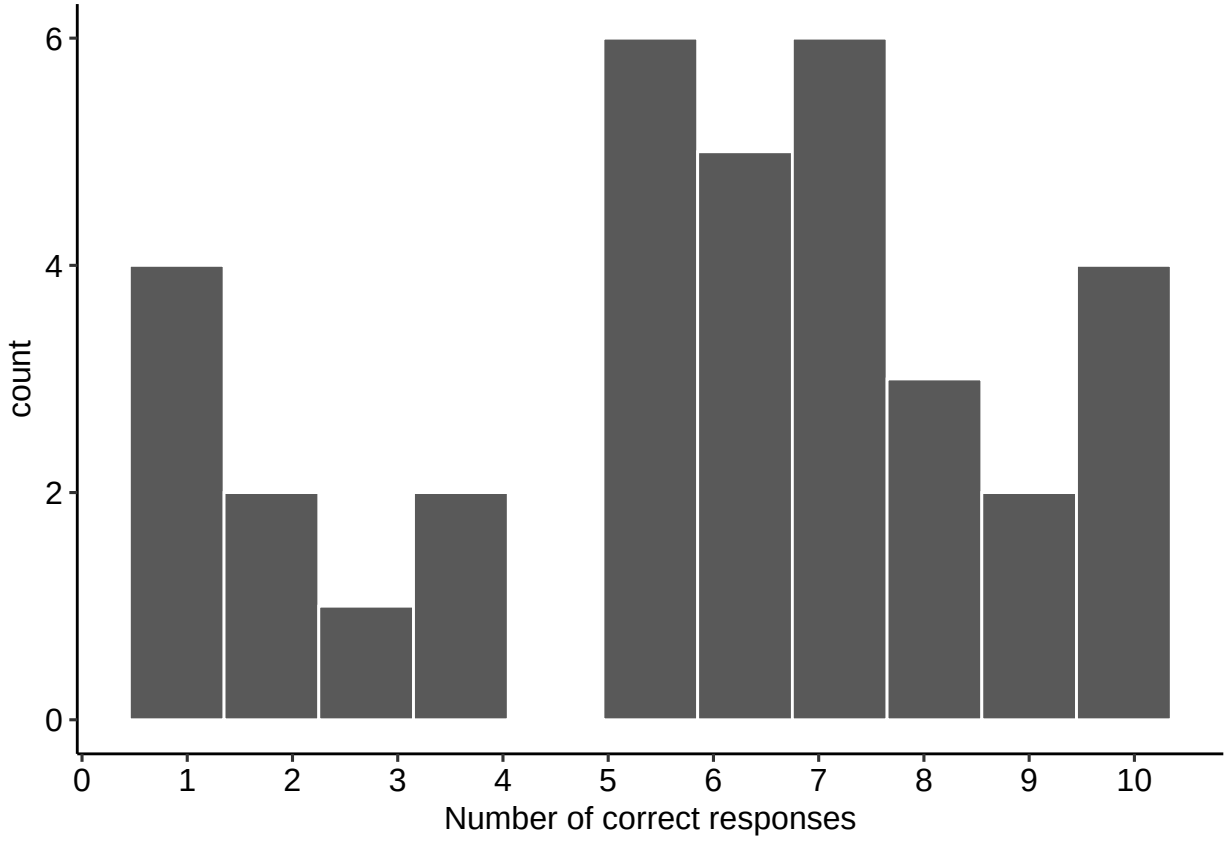

Figure 4: Correct responses on the memory task

```r
data %>%
  group_by(mem_condition) %>%
  summarise(
    m = mean(memory),
    s = sd(memory),
    min = min(memory),
    max = max(memory),
    n = n()
  ) %>%
  kable(booktabs = T,
        col.names = c("Condition", "Mean", "SD", "Min", "Max", "N"),
        digits = c(0, 2, 2, 1, 1, 1),
        caption = "Memory responses by condition") %>%
  kable_styling()
```

Table 1: Memory responses by condition

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

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.

```
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)

data = inner_join(data, mem2)
```

```
data %>%
  ggplot(aes(x = delayed_memory)) +
  geom_histogram(color = "white", bins = 11) +
  scale_x_continuous("Number correct", breaks = c(0:10)) +
  labs(y = "Number of participants") +
  theme_pubr()
```

```
data %>%
  ggplot(aes(x = memory, y = delayed_memory)) +
  geom_point() +
```

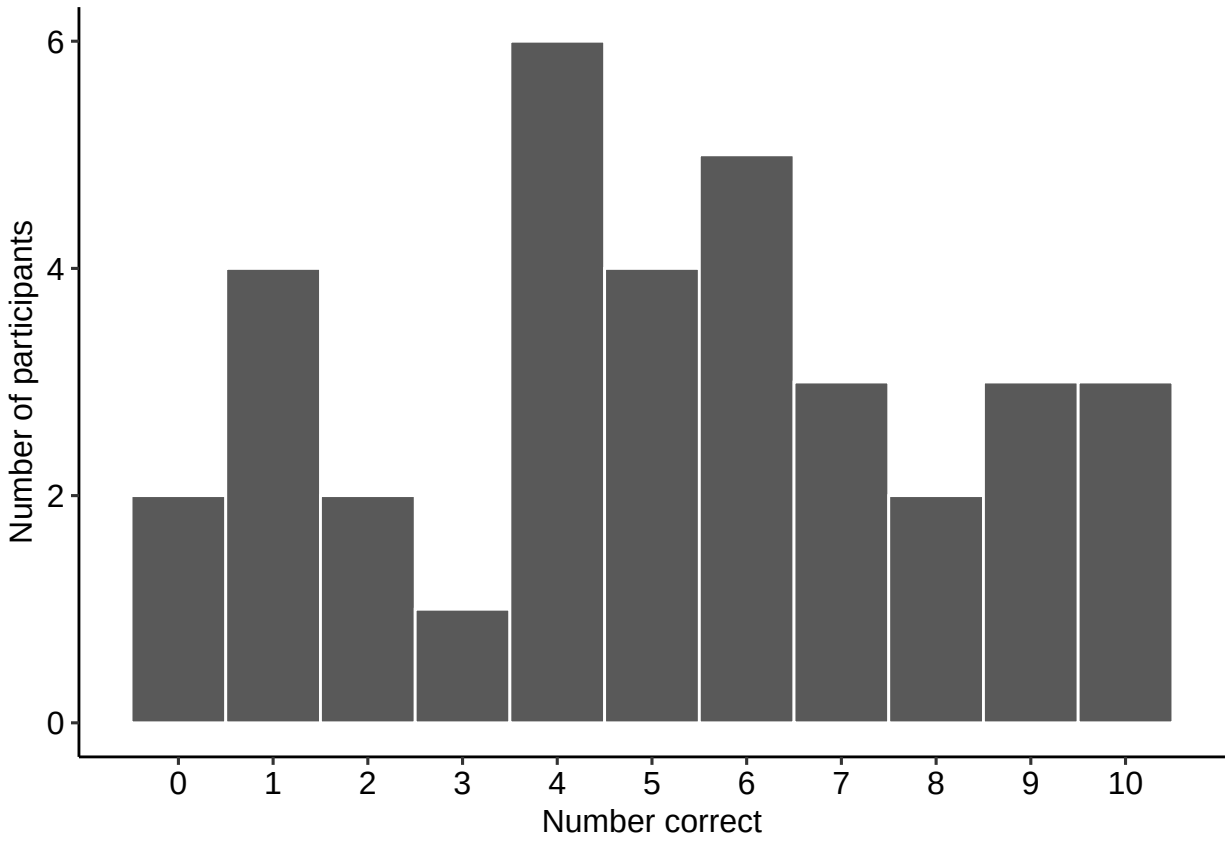

Figure 5: Distribution of delayed memory scores

```
  geom_smooth(method = "lm") +
  scale_x_continuous("Immediate number correct", breaks = c(0:10)) +
  scale_y_continuous("Delayed number correct", breaks = c(0:10)) +
  labs(title = paste0("r = ", printnum(cor(data$memory, data$delayed_memory, use = "pairwise")))) +
  theme_pubr()
```

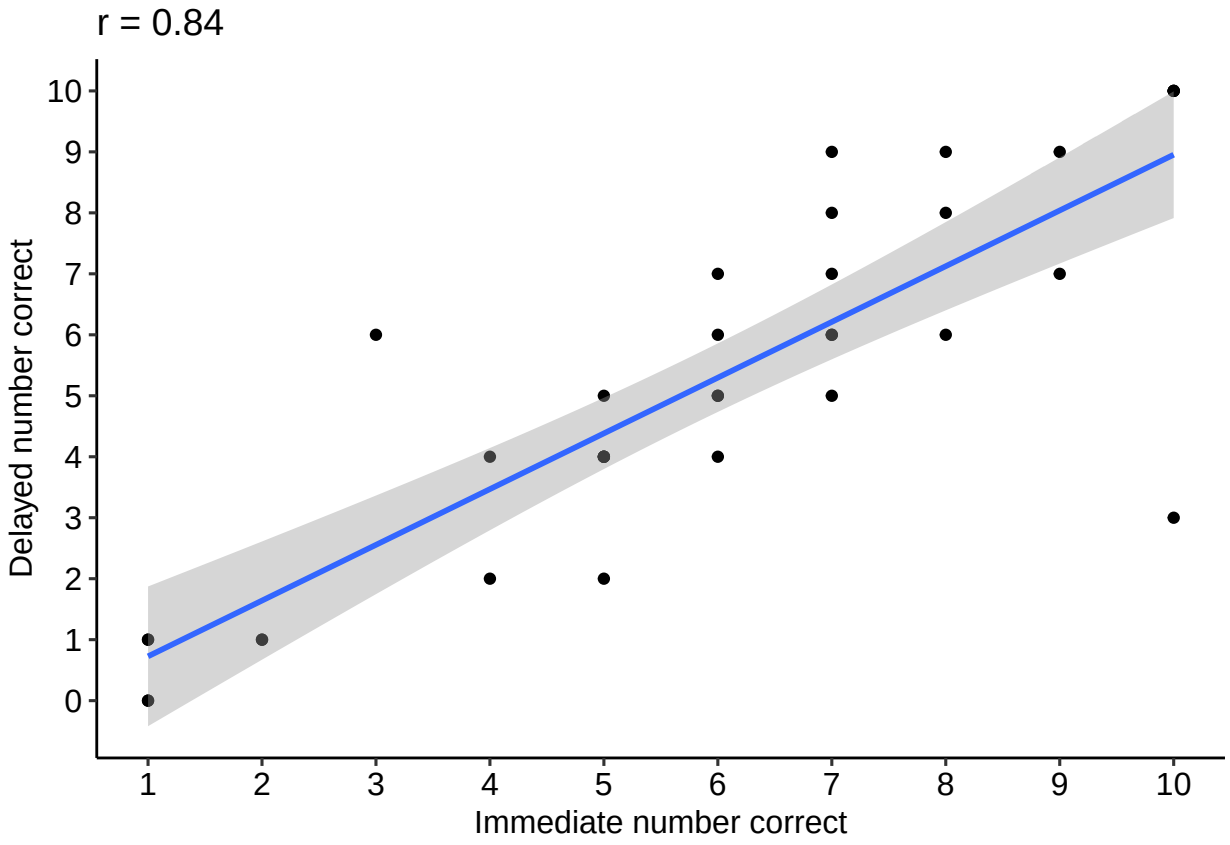

Figure 6: Relationship between immediate and delayed recall

Save data

Does item format affect response?

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

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

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

Workspace

```
library(here) # for working with files
library(tidyverse) # for cleaning
library(janitor) # for variable names
library(lme4) # for mulitlevel modeling
library(sjPlot) # for figures
library(ggpubr) # for prettier plots
library(kableExtra) # for nicer tables
```

Data prep

We will use between-person analyses to compare responses using group-level data for the different formats.

First we select the responses to the items of different formats. These variable names all have the same format: `[trait]_[abcd]` (for example, `talkative_a`). We search for these items using regular expressions.

```
items_seen_first = str_subset(
  names(data),
  "^([[:alpha:]])+_[abcd]$"
)

item_responses = data %>%
  select(proid, all_of(items_seen_first), memory)
```

Next we reshape these data into long form.

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

Response by Format

We used a multilevel model, nesting response within participant to account for dependence. Our primary predictor was format.

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

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

```
## Analysis of Variance Table
##        npar Sum Sq Mean Sq F value
## format    3  2.139 0.71299  0.4306
```

```
#look up p-value
pf(0.23, df1 = 3, df2 = 960, lower.tail = F)
```

```
## [1] 0.8755274
```

```
summary(mod.format)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: response ~ format + (1 | proid)
##    Data: item_responses
##
## REML criterion at convergence: 3680.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.3415 -0.5281  0.2331  0.7215  1.5694
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  proid    (Intercept) 0.1779   0.4218
##  Residual             1.6558   1.2868
## Number of obs: 1085, groups:  proid, 35
##
## Fixed effects:
##                                                Estimate Std. Error t value
## (Intercept)                                     4.66569    0.14501  32.175
## formatAm\nAdjective                            -0.21049    0.21617  -0.974
## formatTend to be\nAdjective                    -0.10521    0.22347  -0.471
## formatI am someone\nwho tends to be\nAdjective  0.02095    0.23253   0.090
##
## Correlation of Fixed Effects:
##             (Intr) frmtAA frTtbA
## frmtAmAdjct -0.671
## frmtTndtbAd -0.649  0.435
## frmIaswttbA -0.624  0.418  0.405
```

```
# mod.format2 = aov(response~format + Error(proid) , data = item_responses)
# summary(mod.format2)
```

```
plot1 = plot_model(mod.format, type = "pred")

plot1$format +
  labs(x = NULL,
       y = "Average response",
       title = "Average responses by item formatting") +
  theme_pubclean()
```

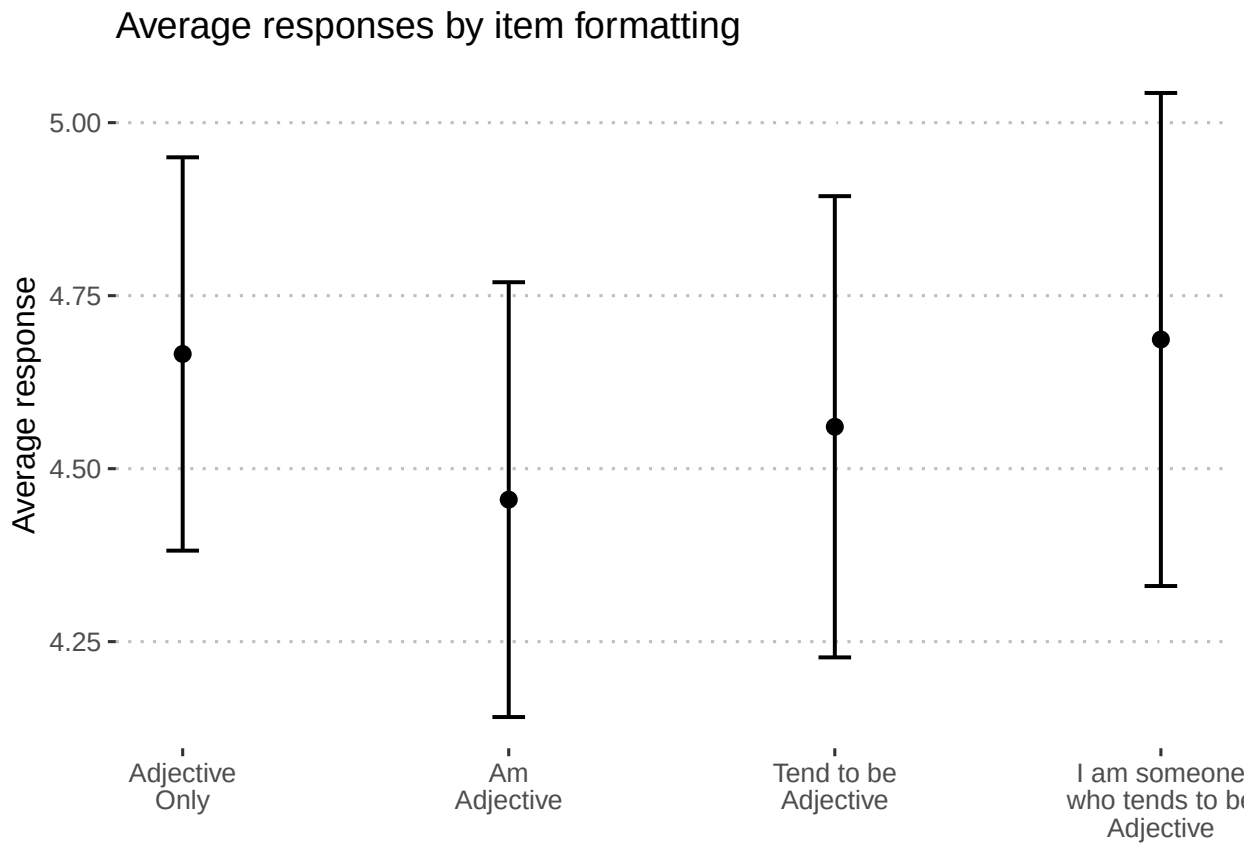

Figure 7: Predicted response on personality items by condition.

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

item_responses %>%
  ggplot(aes(x = response, fill = format)) +
  geom_histogram(bins = 6, color = "white") +
  geom_vline(aes(xintercept = m), data = means_by_group) +
  geom_text(aes(x = 1,
                y = 125,
```

```
              label = paste("M =", round(m,2),
                            "\nSD =", round(s,2))),
            data = means_by_group,
            hjust =0,
            vjust = 1) +
  facet_wrap(~format) +
  guides(fill = F) +
  scale_x_continuous(breaks = 1:6) +
  labs(y = "Number of particpants",
       title = "Distribution of responses by format") +
  theme_pubr()
```

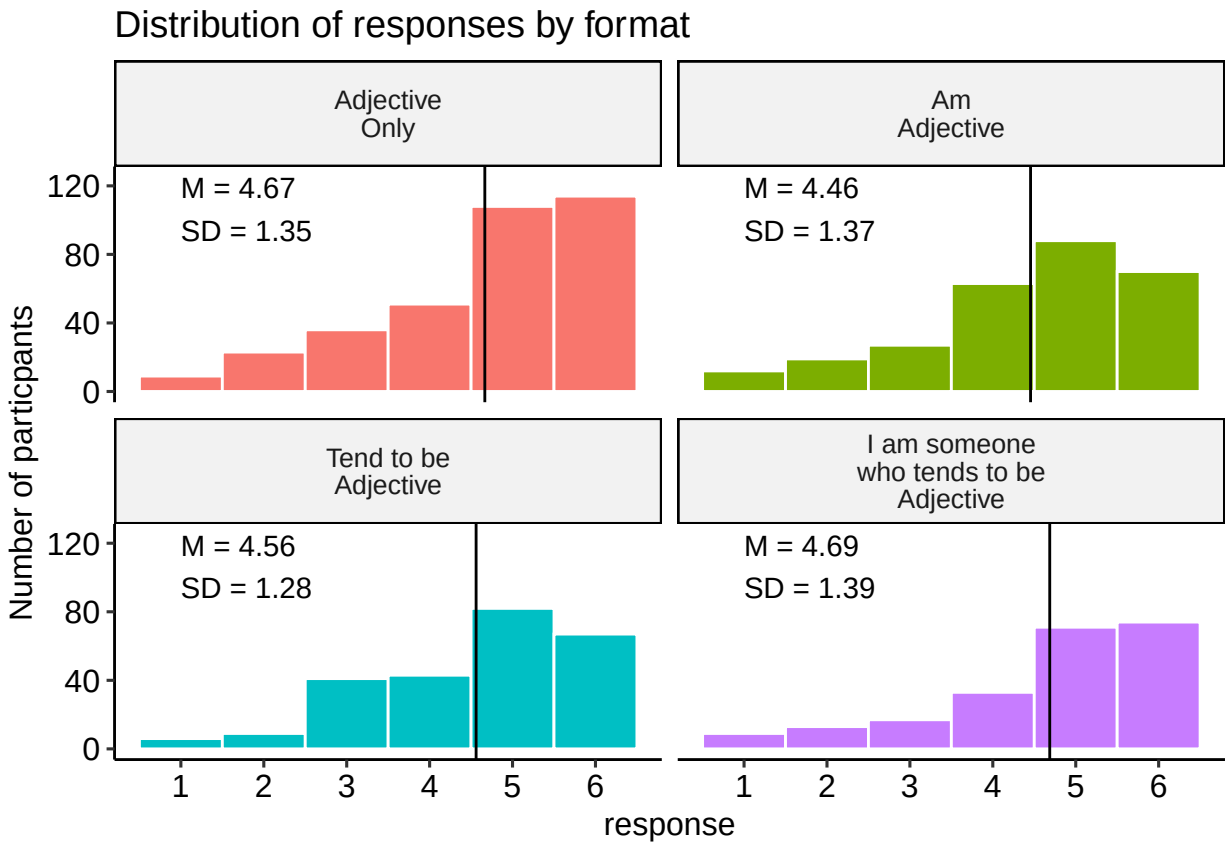

Figure 8: Distribution of responses by category

We can also repeat this analysis separately for each trait.

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

mod_by_item %>%
  mutate(tidy = map(aov, broom::tidy)) %>%
  select(item, tidy) %>%
  unnest(cols = c(tidy)) %>%
```

item	sumsq	meansq	statistic	p.value	p.adj
outgoing	1.58	0.53	0.40	0.76	0.76
helpful	0.20	0.07	0.08	0.97	0.97
reckless	5.14	1.71	0.65	0.59	0.59
moody	6.21	2.07	1.14	0.35	0.35
organized	2.20	0.73	0.66	0.58	0.58
friendly	0.52	0.17	0.34	0.80	0.80
warm	0.71	0.24	0.20	0.90	0.90
worrying	8.95	2.98	1.25	0.31	0.31
responsible	0.87	0.29	0.65	0.59	0.59
lively	1.15	0.38	0.36	0.78	0.78
asleep	0.89	0.30	0.13	0.94	0.94
caring	2.47	0.82	1.17	0.34	0.34
nervous	1.54	0.51	0.20	0.89	0.89
creative	2.35	0.78	0.79	0.51	0.51
hardworking	0.03	0.01	0.02	1.00	1.00
imaginative	1.40	0.47	0.58	0.63	0.63
softhearted	1.25	0.42	0.30	0.83	0.83
calm	3.77	1.26	1.29	0.30	0.30
intelligent	1.86	0.62	0.44	0.73	0.73
curious	0.52	0.17	0.15	0.93	0.93
active	2.80	0.93	0.61	0.61	0.61
human	4.25	1.42	3.03	0.04	0.04
careless	18.23	6.08	2.77	0.06	0.06
impulsive	2.65	0.88	0.45	0.72	0.72
sympathetic	3.42	1.14	1.10	0.36	0.36
cautious	1.55	0.52	0.32	0.81	0.81
talkative	18.53	6.18	3.19	0.04	0.04
sophisticated	1.08	0.36	0.21	0.89	0.89
adventurous	1.34	0.45	0.50	0.68	0.68
thorough	0.33	0.11	0.08	0.97	0.97
thrifty	8.09	2.70	2.06	0.13	0.13

```
  filter(term == "format") %>%
  select(-term, -df) %>%
  mutate(p.adj = p.adjust(p.value, method = "holm")) %>%
  mutate(across(
    starts_with("p"),
    papaja::printnum
  )) %>%
  kable(digits = 2, booktabs = T) %>%
  kable_styling()
```

Response by Format + Memory

```
mod.format_mem = lmer(response~format + memory + (1|proid),
                  data = item_responses)
```

```r
anova(mod.format_mem)
```

```
## Analysis of Variance Table
##        npar Sum Sq Mean Sq F value
## format    3 2.1996  0.7332  0.4428
## memory    1 3.1107  3.1107  1.8787
```

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

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: response ~ format + memory + (1 | proid)
##    Data: item_responses
##
## REML criterion at convergence: 3684
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.3168 -0.5233  0.2396  0.7158  1.5598
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  proid    (Intercept) 0.1715   0.4141
##  Residual             1.6558   1.2868
## Number of obs: 1085, groups:  proid, 35
##
## Fixed effects:
##                                                Estimate Std. Error t value
## (Intercept)                                     4.42246    0.22790  19.405
## formatAm\nAdjective                            -0.22338    0.21337  -1.047
## formatTend to be\nAdjective                    -0.07808    0.22126  -0.353
## formatI am someone\nwho tends to be\nAdjective  0.02897    0.22938   0.126
## memory                                          0.04116    0.03003   1.371
##
## Correlation of Fixed Effects:
##             (Intr) frmtAA frTtbA faswttbA
## frmtAmAdjct -0.386
## frmtTndtbAd -0.475  0.429
## frmIaswttbA -0.411  0.417  0.405
## memory      -0.779 -0.044  0.089  0.026
```

```r
plot_model(mod.format_mem, type = "pred", term = c("format"))
```

```r
plot_model(mod.format_mem, type = "pred", term = c("memory"))
```

```r
mod_by_item = item_responses %>%
  group_by(item) %>%
  nest() %>%
  mutate(mod = map(data, ~lm(response~format + memory, data = .))) %>%
  mutate(aov = map(mod, anova))
```

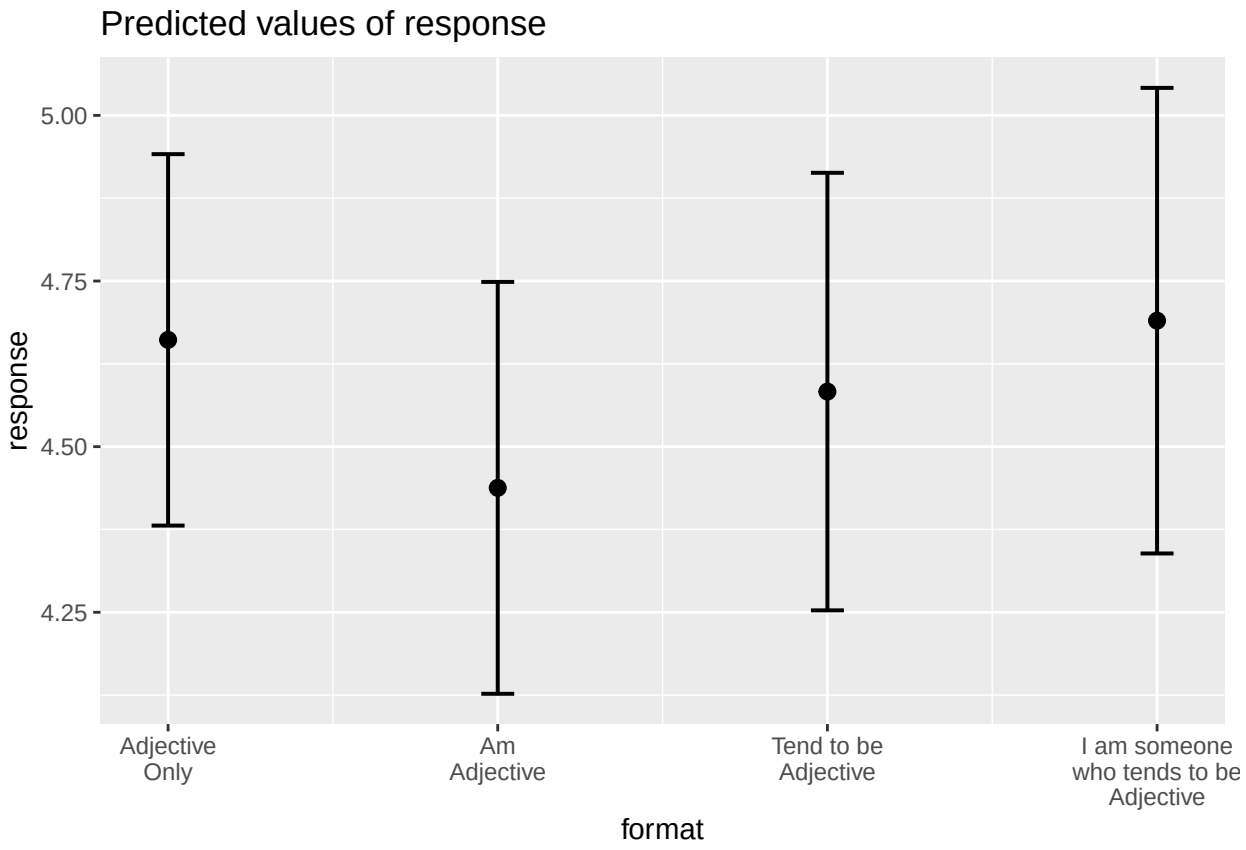

Figure 9: Predicted response on personality items by condition after controlling for memory.

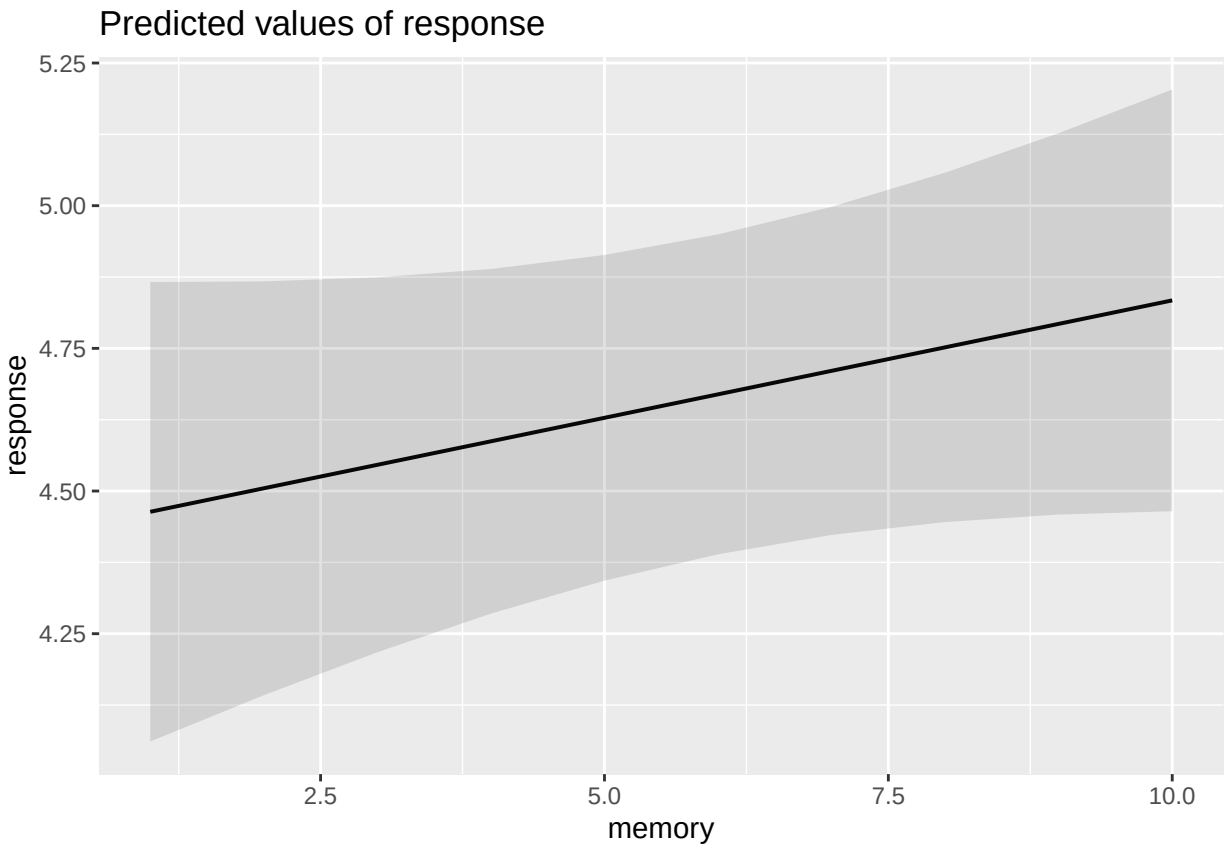

Figure 10: Predicted response on personality items by memory.

item	sumsq	meansq	statistic	p.value	p.adj
outgoing	1.58	0.53	0.39	0.76	0.76
helpful	0.20	0.07	0.08	0.97	0.97
reckless	5.14	1.71	0.63	0.60	0.60
moody	6.21	2.07	1.16	0.34	0.34
organized	2.20	0.73	0.68	0.57	0.57
friendly	0.52	0.17	0.36	0.79	0.79
warm	0.71	0.24	0.24	0.87	0.87
worrying	8.95	2.98	1.21	0.32	0.32
responsible	0.87	0.29	0.63	0.60	0.60
lively	1.15	0.38	0.35	0.79	0.79
asleep	0.89	0.30	0.13	0.94	0.94
caring	2.47	0.82	1.16	0.34	0.34
nervous	1.54	0.51	0.20	0.90	0.90
creative	2.35	0.78	0.81	0.50	0.50
hardworking	0.03	0.01	0.02	1.00	1.00
imaginative	1.40	0.47	0.60	0.62	0.62
softhearted	1.25	0.42	0.30	0.83	0.83
calm	3.77	1.26	1.30	0.29	0.29
intelligent	1.86	0.62	0.44	0.72	0.72
curious	0.52	0.17	0.15	0.93	0.93
active	2.80	0.93	0.60	0.62	0.62
human	4.25	1.42	2.94	0.05	0.05
careless	18.23	6.08	2.87	0.05	0.05
impulsive	2.65	0.88	0.44	0.73	0.73
sympathetic	3.42	1.14	1.08	0.37	0.37
cautious	1.55	0.52	0.31	0.81	0.81
talkative	18.53	6.18	3.10	0.04	0.04
sophisticated	1.08	0.36	0.21	0.89	0.89
adventurous	1.34	0.45	0.49	0.69	0.69
thorough	0.33	0.11	0.09	0.97	0.97
thrifty	8.09	2.70	1.99	0.14	0.14

```
mod_by_item %>%
  mutate(tidy = map(aov, broom::tidy)) %>%
  select(item, tidy) %>%
  unnest(cols = c(tidy)) %>%
  filter(term == "format") %>%
  select(-term, -df) %>%
  mutate(p.adj = p.adjust(p.value, method = "holm")) %>%
  mutate(across(
    starts_with("p"),
    papaja::printnum
  )) %>%
  kable(digits = 2, booktabs = T) %>%
  kable_styling()
```

Questions

- I only used responses in Block 1 here. Should I merge the two blocks together? Or I can repeat the analyses using Block 2? One thing to consider if we merge is that I can't use the simple ANOVA for the item-level analyses (we'll have two responses per person, so back to a nested design).

How is variance in response attributable to participant, adjective, and format?

Within-person analyses will model the proportions of variance attributable to item format, stems of the items (i.e., the content of the adjectives), and the respondent-level variance.

Workspace

```
library(here) # for working with files
library(tidyverse) # for cleaning
library(janitor) # for variable names
library(lme4) # for mulitlevel modeling
library(sjPlot) # for figures
library(ggpubr) # for prettier plots
library(kableExtra) # for nicer tables
```

Data prep

First we select the responses to the items of different formats. These variable names all have the same format: `[trait]_[abcd]` (for example, `talkative_a`). We search for these items using regular expressions.

```
personality_items = str_subset(
  names(data),
  "^([[:alpha:]])+_[abcd]_?2?$"
)

item_responses = data %>%
  select(proid, all_of(personality_items), memory)
```

Next we reshape these data into long form.

```
item_responses = item_responses %>%
  gather(item, response, -proid, -memory) %>%
  mutate(
    time = ifelse(str_detect(item, "_2"), "2", "1"),
    item = str_remove(item, "_2")) %>%
  separate(item, into = c("item", "format")) %>%
  filter(!is.na(response))
```

Model

We estimate variance attributable to particpant (`proid`), adjective (`item`), and format (`format`) using a nested model.

```
mod_within_full = lmer(response ~ 1
                  + (1 | item)
                  + (1 | format)
                  + (1 |proid),
                  data = item_responses)
summary(mod_within_full)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: response ~ 1 + (1 | item) + (1 | format) + (1 | proid)
##    Data: item_responses
##
## REML criterion at convergence: 6724.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.1453 -0.5052  0.0946  0.6176  3.1789
##
## Random effects:
##  Groups   Name        Variance  Std.Dev.
##  proid    (Intercept) 0.1622567 0.40281
##  item     (Intercept) 0.5149790 0.71762
##  format   (Intercept) 0.0005179 0.02276
##  Residual             1.1924370 1.09199
## Number of obs: 2170, groups:  proid, 35; item, 31; format, 4
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   4.5802     0.1481   30.93
```

```
variances = VarCorr(mod_within_full, comp="Variance")
var_proid  = variances$proid[[1]]
var_item   = variances$item[[1]]
var_format = variances$format[[1]]
var_resid = attr(variances, "sc")^2
var_total = var_proid + var_item + var_format + var_resid
```

Participants account for 8.68 percent of the variability in response.

Items account for 27.54 percent of the variability in response.

Format accounts for 0.03 percent of the variability in response.

In total, 36.24 percent of the variability in response is explained.