Supplemental file

Last updated 2021-07-14

Contents

Cleaning

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
library(stringdist)
```

```
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()
```

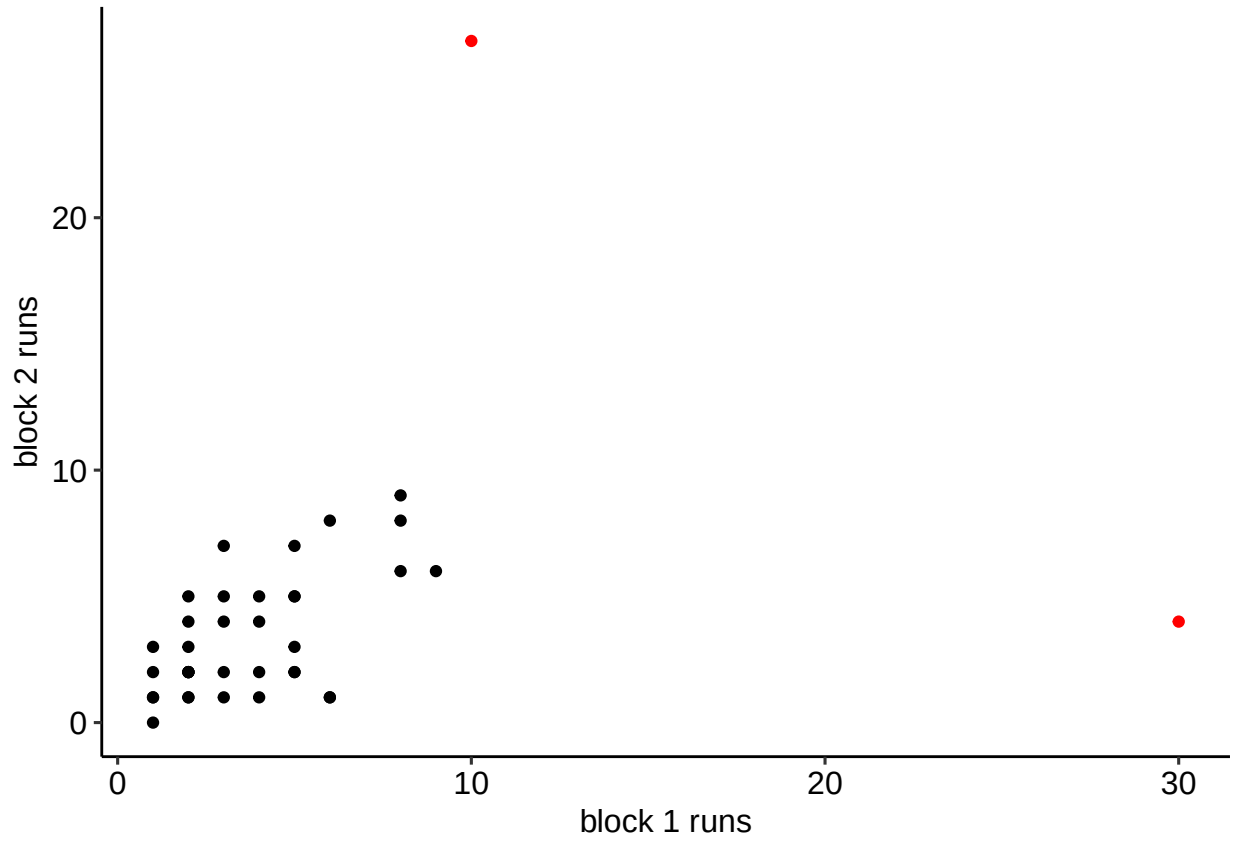


Figure 1: Maximum number of same consecutive responses in personality blocks.

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)) %>%
  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()
```

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

First we count how many answers each participant responded to. Why count using code and not just looking at qualtrics? Two reasons: 1. It's not entirely clear from Qualtrics how many questions are in the survey, because there are 4 versions of each trait item and 4 different recall conditoins. 2. We expect the survey to change after reviews during Stage 1, and the person who wrote this code will definitely forget to change this section later.

```
num_variables = data %>%
  select(-c(start_date:response_id),
         -starts_with("t_"),
         -any_of(c("prolific_pid")) # not used. will we clean it up later? doesn't matter!
  ) %>%
```

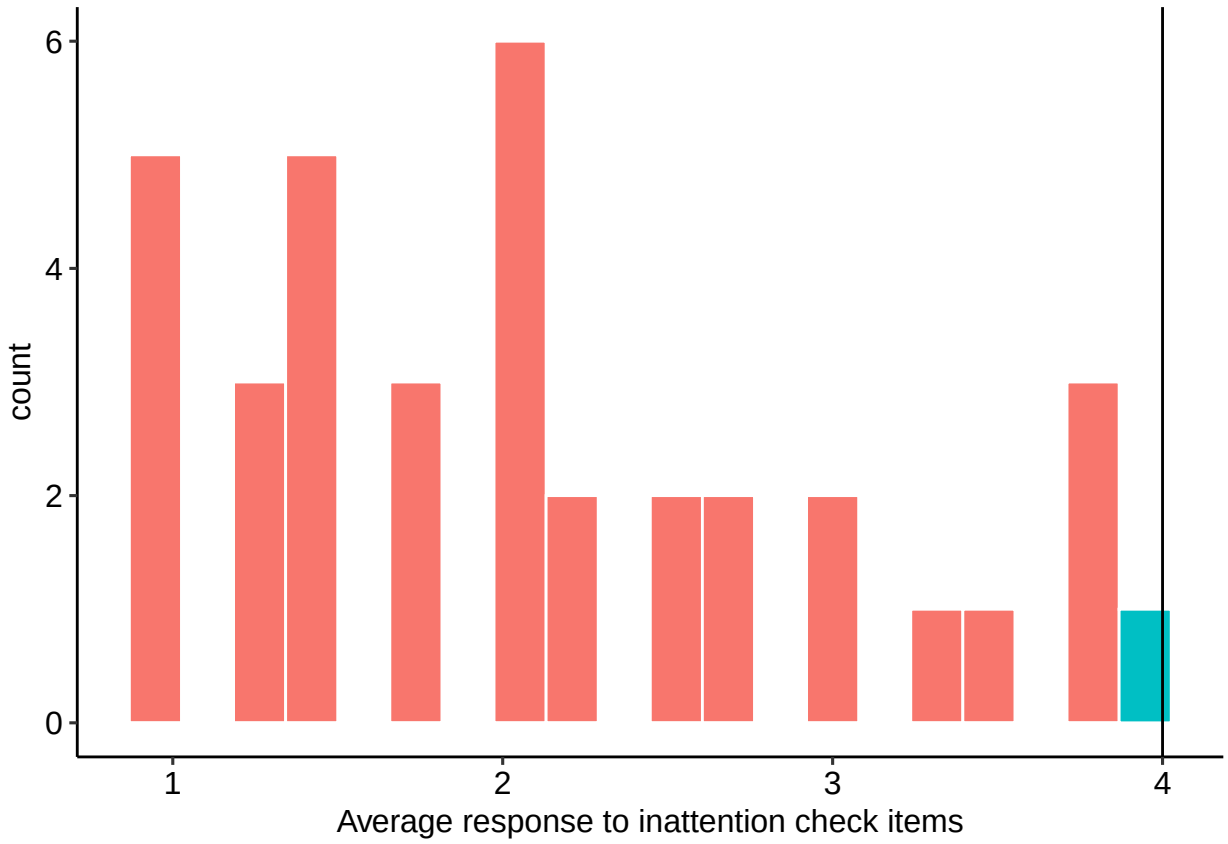


Figure 2: Average response to inattention check items

```
  gather(item,response, -proid) %>%
  filter(!is.na(response)) %>%
  group_by(proid) %>%
  count(name = "num_var") #how many variables per person?

# should be about the same number of responses per person with forced response
table(num_variables$num_var)
```

```
##
## 82 83
## 32  3
```

```
# participants with one more response included text in the response to the question
# Why should we exclude you?
```

Now we calculate the average time to respond by taking the duration in seconds variable and dividing by the number of responses.

```
num_variables = data %>%
  select(proid, duration_in_seconds) %>%
  full_join(num_variables) %>%
  mutate(time_per_item = duration_in_seconds/num_var) %>%
  mutate(remove = case_when(
    time_per_item < 1 ~ "Remove",
    time_per_item > 30 ~ "Remove",
    TRUE ~ "Keep"
  ))
```

```
num_variables %>%
  ggplot(aes(x = time_per_item, fill = remove)) +
  geom_histogram(bins = 30, color = "white") +
  labs(x = "Seconds per item") +
  theme_pubr()
```

We remove 1 participants whose average item response time is outside the allowed range.

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

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(
```

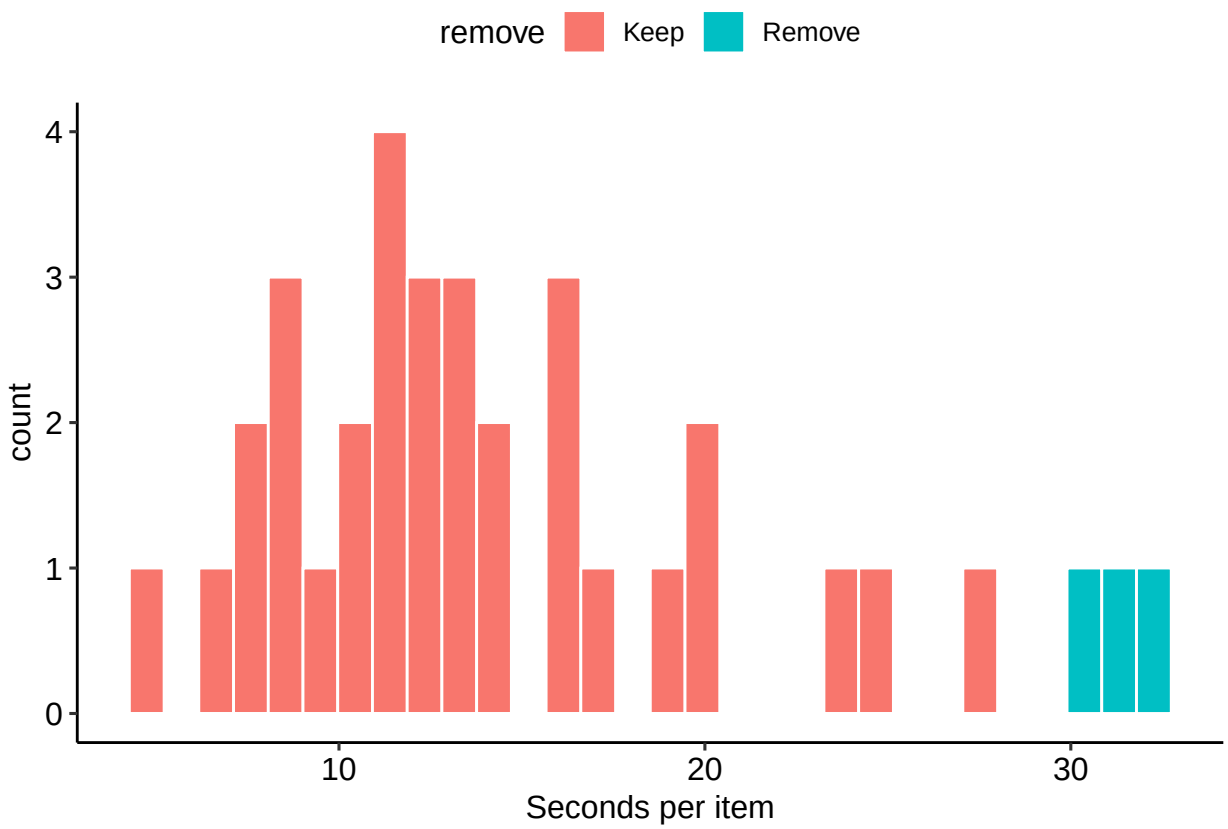


Figure 3: Average item response time

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

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 5.59 words correctly ($SD = 2.76$),

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

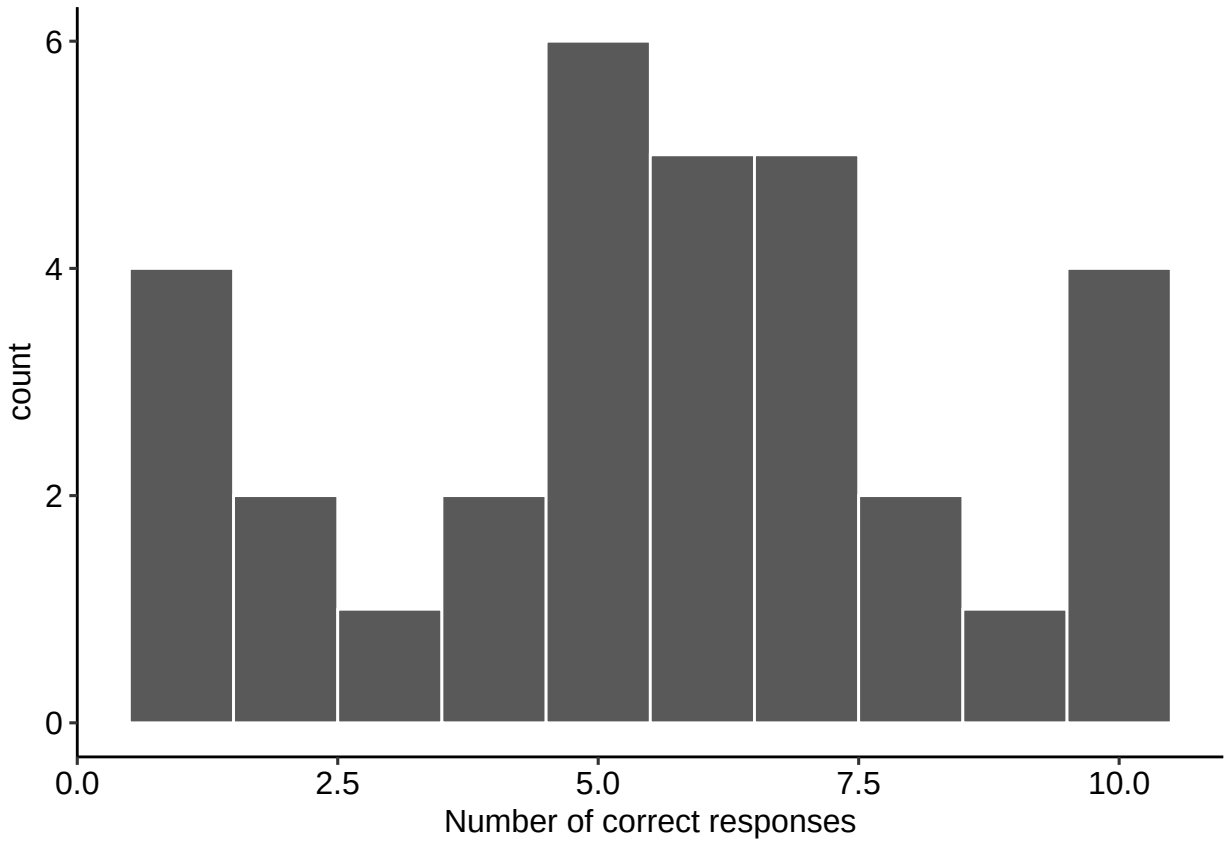


Figure 4: Correct responses on the memory task

Table 1: Memory responses by condition

Condition	Mean	SD	Min	Max	N
1	5.50	2.56	1	10	8
2	4.29	3.30	1	10	7
3	6.11	2.71	1	10	9
4	6.25	2.60	2	10	8

```
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()
```

Does item format affect response?

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))
```

Next we reshape these data into long form.

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

Results

```
item_responses$format = as.factor(item_responses$format)
item_responses$format = relevel(item_responses$format, ref = "a")

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 1.1015 0.36715    0.23
```

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

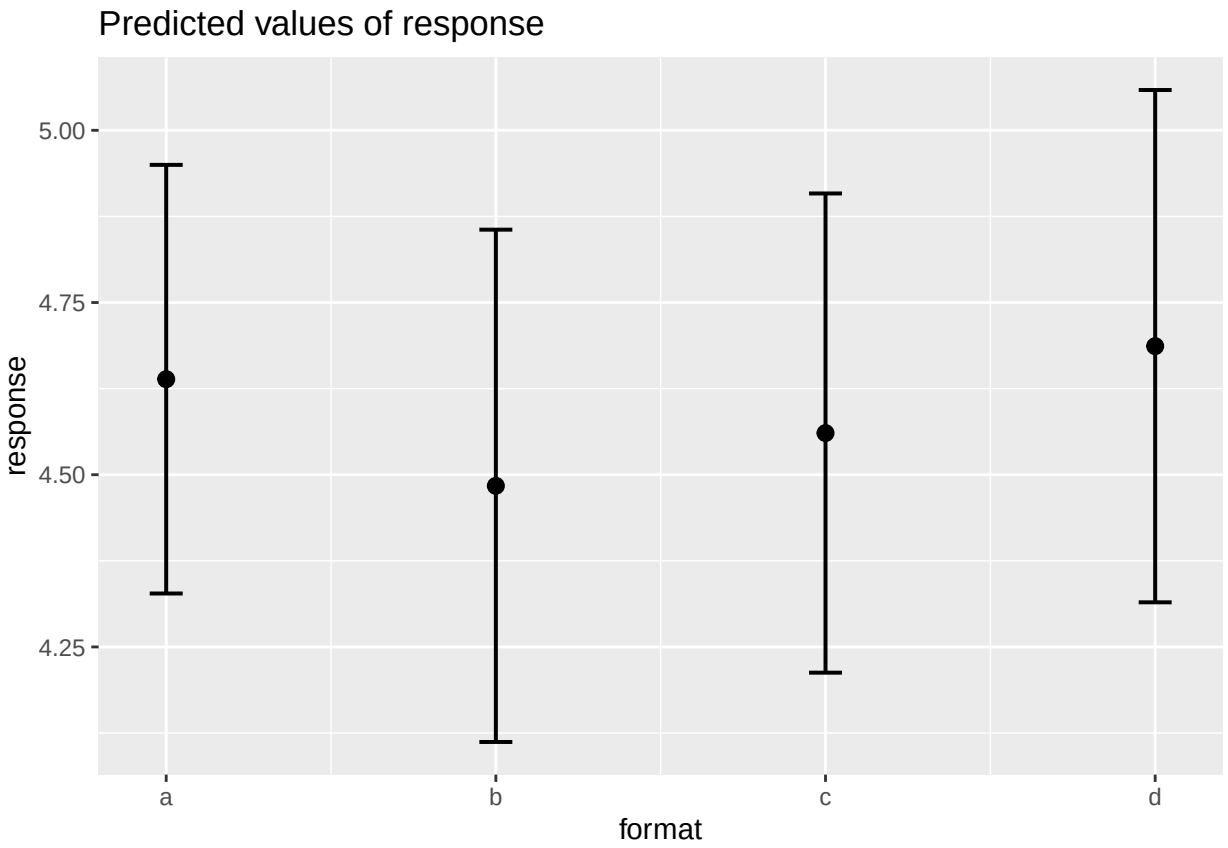
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: response ~ format + (1 | proid)
##    Data: item_responses
##
## REML criterion at convergence: 3332.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.4196 -0.5368  0.2336  0.7280  1.6140
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  proid    (Intercept) 0.2005   0.4478
##  Residual             1.5966   1.2636
## Number of obs: 992, groups:  proid, 32
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  4.63871    0.15875  29.221
## formatb     -0.15484    0.24739  -0.626
## formatc     -0.07823    0.23812  -0.329
## formatd      0.04793    0.24739   0.194
##
## Correlation of Fixed Effects:
##         (Intr) formtb formtc
## formatb -0.642
## formatc -0.667  0.428
## formatd -0.642  0.412  0.428
```

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

```
##
## Error: proid
##           Df Sum Sq Mean Sq F value Pr(>F)
## format     3   5.39   1.796    0.23  0.875
## Residuals 28 218.74   7.812
##
## Error: Within
##            Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 960   1533   1.597
```

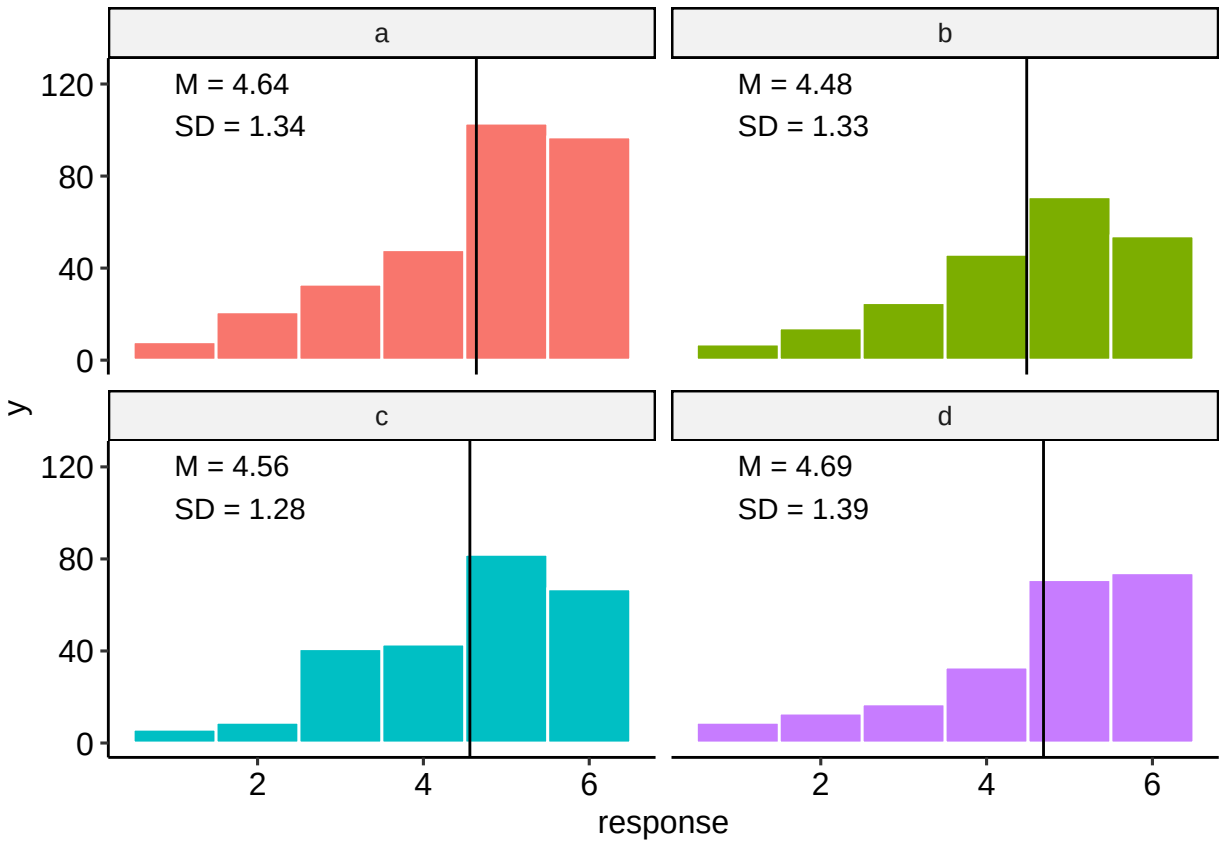
```
plot_model(mod.format, type = "pred")
```

```
## $format
```




```
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) +
  theme_pubr()
```



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

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).

item	term	df	sumsq	meansq	statistic	p.value	p.adj
outgoing	format	3	1.24	0.41	0.29	0.83	0.83
helpful	format	3	0.86	0.29	0.36	0.78	0.78
reckless	format	3	5.39	1.80	0.80	0.50	0.50
moody	format	3	2.44	0.81	0.45	0.72	0.72
organized	format	3	2.55	0.85	0.75	0.53	0.53
friendly	format	3	0.36	0.12	0.23	0.88	0.88
warm	format	3	1.53	0.51	0.42	0.74	0.74
worrying	format	3	5.89	1.96	0.81	0.50	0.50
responsible	format	3	0.43	0.14	0.34	0.79	0.79
lively	format	3	0.47	0.16	0.14	0.93	0.93
asleep	format	3	1.16	0.39	0.20	0.90	0.90
caring	format	3	1.55	0.52	0.70	0.56	0.56
nervous	format	3	0.34	0.11	0.05	0.98	0.98
creative	format	3	1.83	0.61	0.58	0.63	0.63
hardworking	format	3	0.02	0.01	0.01	1.00	1.00
imaginative	format	3	1.93	0.64	0.83	0.49	0.49
softhearted	format	3	0.87	0.29	0.20	0.90	0.90
calm	format	3	3.35	1.12	1.13	0.35	0.35
intelligent	format	3	1.96	0.65	0.44	0.73	0.73
curious	format	3	0.81	0.27	0.24	0.87	0.87
active	format	3	2.39	0.80	0.49	0.69	0.69
human	format	3	3.70	1.23	2.84	0.06	0.06
careless	format	3	8.74	2.91	1.42	0.26	0.26
impulsive	format	3	2.28	0.76	0.36	0.78	0.78
sympathetic	format	3	4.05	1.35	1.27	0.31	0.31
cautious	format	3	1.44	0.48	0.29	0.84	0.84
talkative	format	3	13.60	4.53	2.27	0.10	0.10
sophisticated	format	3	0.27	0.09	0.06	0.98	0.98
adventurous	format	3	1.15	0.38	0.43	0.74	0.74
thorough	format	3	0.38	0.13	0.10	0.96	0.96
thrifty	format	3	8.89	2.96	2.22	0.11	0.11