Winter 2022 SGC 4A Data Cleaning

Amy Rae Fox

04/07/2022

Contents

The purpose of this file is processing the combined data files for Winter 2022 into files that contain only valid data for analysis, excluding invalid sessions and participants

IMPORTANT
Must manually define working directory at 1 level above 4A [ie at 2022_winter level]. This is required for splitting and saving 4B control participants.

Data is imported from 2 files, indicating two levels of analysis: participants and blocks (item-level).

Note: mouse-cursor data contained in final_mouse_blocks.json file is not handled here.

```
#IMPORT DATA
df_participants <- fromJSON("input/winter22_sgc4a_final_participants.json")
df_items <- fromJSON('input/winter22_sgc4a_final_items.json')

#add term indicator
df_participants$term <- "winter22"
df_items$term <- "winter22"

#DEFINE SGC_4A validity crieria
sessions <- c('wi22sona') #SGC4A second online replication on SONA
conditions <-c(11111,113,114,115) #4 conditions
violation_threshold = 3 #number of allowable browser violations
effort_exclusion = c("I didn't try very hard, or rushed through the questions", "I started out trying hard
n_items = 15 #fifteen items is complete dataset per participant
```

```
#placeholder for excluding participants
ex_participants = data.frame()
```

note : We drop all scores calculated in the stimulus engine (except absolute score, which uses simple # strictly correct), as they are recalculate during analysis using a different MC scoring algorithm.

```
#create factors in PARTICIPANTS
df_participants <- df_participants %>%
  mutate( #create factors and remove extraneous ""
    subject=factor(subject),
    condition=factor(condition),
    pretty_condition = recode_factor(condition, "11111" = "Orth-Full", "114" =  "Orth-Sparse", "115" = "Or
    study = factor(study),
    condition = factor(condition),
    session = factor(session),
    exp_id = factor(exp_id),
    sona_id = factor(sona_id),
    pool = factor(pool),
    mode = factor(mode),
    attn_check = factor(attn_check),
    status=factor(status),
    term=factor(term),
    gender = as.factor(gender),
    age = as.integer(age),
    country = gsub('"',"",country),
    year = factor(schoolyear),
    major = factor(major),
    browser = factor(browser),
    os = factor(os),
    native_language = factor(language),
    totaltime_m = totaltime/1000/60,
   ) %>% select( #order cols
    subject,
    study,
    condition,
    pretty_condition,
    session,
    exp_id,
    sona_id,
    pool,
    mode,
    attn_check,
    explanation,
    effort,
    difficulty,
    confidence,
    enjoyment,
    other,
    age,
    country,
    language,
    schoolyear,
    major,
    gender,
    disability,
```

```
    browser,
    width,
    height,
    os,
    starttime,
    status,
    term,
    violations,
    absolute_score,
    # discriminant_score,
    # tri_score,
    # orth_score,
    # other_score,
    # blank_score,
    totaltime_m
   )
```

```
df_items <- df_items %>%
  mutate(
    subject=factor(subject),
    condition=factor(condition),
    pretty_condition = recode_factor(condition, "11111" = "Orth-Full", "114" =  "Orth-Sparse", "115" = "Or
    pool=factor(pool),
    mode = factor(mode),
    explicit=factor(explicit),
    impasse = factor(impasse),
    grid = factor(grid),
    mark = factor(mark),
    ixn = factor(ixn),
    term=factor(term),
    relation = factor(relation),
    block = factor(block),
    correct = factor(correct),
    q=factor(q),
    rt_s = rt/1000,
    time_elapsed_m = time_elapsed/1000/60
  ) %>% select(
     subject,
     study,
     term,
     pool,
     mode,
     condition,
     pretty_condition,
     block,
     explicit,
     impasse,
     grid,
     mark,
     ixn,
     gwidth,
     gheight,
     graph,
     time_elapsed_m,
     question,
     relation,
```

```
    q,
    correct,
    # discriminant,
    # tri_score,
    # orth_score,
    # other_score,
    # blank_score,
    answer,
    rt_s,

  )
```

Data Validation

Exclusions

Completion Status

Starting with Winter 2022, data are saved to the database even if the subject's browser did not meet minimum specifications (at which point they are prompted to change browsers, or end the study). This allows us to learn about the browsers, screen sizes and OS that (potential) subjects are using. However, these data are *not* exported from the database for analysis (see flatten.js and status.js scripts). Thus, only subjects who successfully completed the entire study are included in this file.

```
#MANUALLY INSPECT status
df_participants %>% group_by(status) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 1 x 2
##   status      n
##   <fct>   <int>
## 1 success   582
```

582 successfully completed the study.

```
#DISCARD participants from invalid sessions
exclude_status <- df_participants %>%
          filter(status != "success") %>%
          mutate(reason="invalid-status")

ex_participants <- rbind(ex_participants, exclude_status)
rm(exclude_status)

df_participants <- df_participants %>%
  filter( ! subject %in% ex_participants$subject)
```

No data need to be excluded on account of completion status.

Conditions

Participants are randomly assigned to an experimental condition when starting the study. Here we validate that only conditions for the current study are included in this dataset.

```
#MANUALLY INSPECT conditions
df_participants %>% group_by(condition) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 4 x 2
##   condition     n
##   <fct>     <int>
## 1 11111       238
## 2 113         119
## 3 114         105
## 4 115         120
```

Data from conditions *not* corresponding to valid conditions should be discarded.

Split Control Condition

238 subjects were collected under SGC4A condition 11111. However, *half* of these subjects were *actually* SGC4B (also control condition, identical design), but the study code was hardcoded. Thus, we need to *split* and take half the SGC4A 11111 participants, remove them from this dataset, and export them to be moved to SGC4B.

```
#SPLIT PARTICIPANT FILE

#divide df_participants into control and experimental conditions
df_participants_control <- df_participants %>% filter(condition == "11111")
df_participants_experimental <- df_participants %>% filter(condition %in%  c("113","114","115"))

#split control condition participants into two halves
df_participants_keep <- df_participants_control[seq_len(nrow(df_participants_control)) %% 2 == 0, ]    # Ex
df_participants_split <- df_participants_control[seq_len(nrow(df_participants_control)) %% 2 == 1, ]    # E

#validate keep + split = control
nrow(df_participants_control) == nrow(df_participants_keep) + nrow(df_participants_split)
```

```
## [1] TRUE
```

```
#RECONSTRUCT DF_PARTICIPANTS
df_participants <- rbind(df_participants_experimental, df_participants_keep)

#SPLIT ITEMS FILE
df_items_keep <- df_items %>% filter(subject %in% df_participants$subject)
df_items_split <- df_items %>% filter(subject %in% df_participants_split$subject)

#validate all items have been split
nrow(df_items) == nrow(df_items_keep) + nrow(df_items_split)
```

```
## [1] TRUE
```

```
#WRITE SGC4B FILES

#set study = SGC4B
df_participants_split$study <- "SGC4B"
df_items_split$study <- "SGC4B"
```

```
#WRITE 4B CONTROL TO OUTPUT OF THIS STUDY FOR REFERENCE
write.csv(df_participants_split,"output/winter22_sgc4b_CONTROL_participants.csv", row.names = FALSE)
write.csv(df_items_split,"output/winter22_sgc4b_CONTROL_items.csv", row.names = FALSE)

#WRITE 4B CONTROL TO INPUT OF 4B WHERE 4B CLEANING FILE CAN FIND IT
write.csv(df_participants_split,"../SGC4B/input/winter22_sgc4b_CONTROL_participants.csv", row.names = FALSE)
write.csv(df_items_split,"../SGC4B/input/winter22_sgc4b_CONTROL_items.csv", row.names = FALSE)

#RECONSTRUCT DF_ITEMS
df_items <- df_items_keep

#cleanup temp dataframes
rm(df_participants_control,df_participants_experimental, df_participants_keep, df_participants_split, df_i
```

After splitting the extra control condition participants into SGC4, we are left with:

```
#MANUALLY INSPECT conditions
df_participants %>% group_by(condition) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 4 x 2
##   condition     n
##   <fct>     <int>
## 1 11111       119
## 2 113         119
## 3 114         105
## 4 115         120
```

```
#DISCARD participants from conditions invalid for this study
exclude_condition <- df_participants %>%
          filter(!condition %in% conditions) %>%
          mutate(reason="invalid-condition")

ex_participants <- rbind(ex_participants, exclude_condition)
rm(exclude_condition)

df_participants <- df_participants %>%
  filter( ! subject %in% ex_participants$subject)
```

No data need to be excluded on account of condition.

Sessions

The (string) `session code` is embedded in the URL querystring by the experimenter to differentiate testing sessions in SONA from demo and other environment setup tasks.

```
#MANUALLY INSPECT sessions
df_participants %>% group_by(session) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 1 x 2
##   session      n
##   <fct>    <int>
## 1 wi22sona   463
```

Data from sessions not corresponding to valid sessions should be discarded.

```
#DISCARD participants from invalid sessions
exclude_session <- df_participants %>%
          filter(!session %in% sessions) %>%
          mutate(reason="invalid-session")

ex_participants <- rbind(ex_participants, exclude_session)
rm(exclude_session)

df_participants <- df_participants %>%
  filter( ! subject %in% ex_participants$subject)
```

No data need to be excluded on account of session.

Browser Interaction Violations

Browser interaction data is recorded by jspsych allowing us to determine if subjects violate our instructions not to leave the browser tab (or exit fullscreen mode) during test. These incidents are recorded in jspsych interaction data object, and the number of violations is counted and added to the participant data file.

Due to eccentricity of the browser events captured, 1-2 browser violations can be captured even if the subject did not leave the browser window (eg. in case of resizing window to meet minimum requirements.)

```
#MANUALLY INSPECT violations
df_participants %>% group_by(violations) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 15 x 2
##    violations     n
##         <dbl> <int>
##  1        1     298
##  2        1.5    21
##  3        2      81
##  4        2.5     8
##  5        3      23
##  6        3.5     3
##  7        4       8
##  8        4.5     2
##  9        5       9
## 10        5.5     1
## 11        6       3
## 12        6.5     2
## 13        7       1
## 14        8       2
## 15        9       1
```

```
#DISCARD participants exceeding the threshold of browser interaction violations
exclude_violations <- df_participants %>%
          filter(violations > violation_threshold) %>%
          mutate(reason="exceeded-violations")

ex_participants <- rbind(ex_participants, exclude_violations)
```

```
rm(exclude_violations)

df_participants <- df_participants %>%
  filter( ! subject %in% ex_participants$subject)
```

Thirty two participants were excluded for exceeding the maximum allowed number of browser interaction violations.

Effort

To assist in mitigating increased noise in data collected asynchronously from the UCSD student subject pool, we added explicit ratings of how much effort the participant expended on the task. This question was implemented as a multiple-choice drop-down on an 'Effort' page prior to the 'Demographics' survey at the end of the study. Subjects were given four options : (1) I tried my best on each question, (2) I tried my best on most questions, (3) I started out trying hard, but gave up at some point, (4) I didn't try very hard, or rushed through the questions.

```
#MANUALLY INSPECT effort
df_participants %>% group_by(effort) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 4 x 2
##   effort                                                    n
##   <chr>                                                 <int>
## 1 I didn't try very hard, or rushed through the questions     3
## 2 I started out trying hard, but gave up at some point       31
## 3 I tried my best on each question                          264
## 4 I tried my best on most questions                         133
```

Participants answering with options *I didn't try very hard, or rushed through the questions* or *I started out trying hard, but gave up at some point* are excluded from analysis.

```
#DISCARD participants who indicated they did not expend adequate effort on the study
exclude_effort <- df_participants %>%
          filter(effort %in% effort_exclusion) %>%
          mutate(reason="selfrated-effort")

ex_participants <- rbind(ex_participants, exclude_effort)
rm(exclude_effort)

df_participants <- df_participants %>%
  filter( ! subject %in% ex_participants$subject)
```

Thirty-four participants are excluded for low (self-rated) effort.

Attention Check

The 6th question in the study is non-discriminatory (can easily get correct answer regardless of strategy) and serves as an attention check question.

```
#MANUALLY INSPECT attention
df_participants %>% group_by(attn_check) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 2 x 2
##   attn_check     n
##   <fct>      <int>
## 1 FALSE         37
## 2 TRUE         360
```

Participants who answered the attention check question incorrectly should be excluded.

```
#DISCARD participants who indicated they did not expend adequate effort on the study
exclude_attn <- df_participants %>%
          filter(attn_check == FALSE) %>%
          mutate(reason="failed-attnchk")

ex_participants <- rbind(ex_participants, exclude_attn)
rm(exclude_attn)

df_participants <- df_participants %>%
  filter( ! subject %in% ex_participants$subject)
```

Thirty seven participants are excluded for failing the attention check question.

Items

Next, we need to discard item_level data for excluded participants.

```
ex_items <- df_items %>%
  filter (subject %in% ex_participants$subject)

df_items <- df_items %>%
  filter (!subject %in% ex_participants$subject )
```

Validation

After all exclusions, we are left with the following number of participants per condition:

```
#MANUALLY INSPECT conditions
df_participants %>% group_by(condition) %>%
  dplyr::summarize(n=n())
```

```
## # A tibble: 4 x 2
##   condition     n
##   <fct>     <int>
## 1 11111        88
## 2 113          86
## 3 114          88
## 4 115          98
```

Finally, we need to validate we have a complete set of items for all valid participants.

```
count(df_items)[[1]] == count(df_participants)[[1]]* n_items
```

```
## [1] TRUE
```

Participants Codebook

```
#see https://cran.r-project.org/web/packages/codebook/vignettes/codebook_tutorial.html

#ADD VARIABLE METADATA
dict <- rio::import("input/dictionary_sgc4a_participants.csv", "csv") #import data dictionary
var_label(df_participants) <- dict %>% select(VARIABLE, DESCRIPTION) %>% dict_to_list() #add variable labe

#ADD DATASET METATDATA
metadata(df_participants)$name <- "Experimental PARTICIPANTS for study SGC4A"
metadata(df_participants)$description <- "Data for study SGC4A summarized at PARTICIPANT  level"
metadata(df_participants)$creator <- "Amy Rae Fox"
metadata(df_participants)$contact <- "amyraefox@gmail.com"
```

```
#{r, eval = checkMode() == "pdf"} #ONLY FOR PDF KNIT
codebook::skim_codebook(df_participants)
```

Table 1: Data summary

Name	data
Number of rows	360
Number of columns	33
Column type frequency:	
character	8
factor	16
numeric	9
Group variables	None

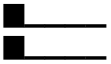
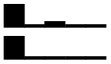
Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
explanation	0	1	0	422	2	359	0
effort	0	1	32	33	0	2	0
other	0	1	0	345	234	115	0
country	0	1	2	25	0	41	0
language	0	1	6	9	0	8	0
schoolyear	0	1	5	7	0	6	0
disability	0	1	0	130	160	39	0
starttime	0	1	24	24	0	360	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
subject	0	1	FALSE	360	075: 1, 07F: 1, 083: 1, 0BT: 1
study	0	1	FALSE	1	SGC: 360
condition	0	1	FALSE	4	115: 98, 111: 88, 114: 88, 113: 86
pretty_condition	0	1	FALSE	4	Ort: 98, Ort: 88, Ort: 88, Tri: 86
session	0	1	FALSE	1	wi2: 360
exp_id	0	1	FALSE	2	221: 301, 221: 59
sona_id	0	1	FALSE	356	347: 3, 319: 2, 368: 2, 268: 1
pool	0	1	FALSE	1	son: 360
mode	0	1	FALSE	1	asy: 360
attn_check	0	1	FALSE	1	TRU: 360, FAL: 0
major	0	1	FALSE	7	Soc: 257, Bio: 40, Hum: 21, Mat: 17
gender	0	1	FALSE	3	Fem: 260, Mal: 93, Oth: 7
browser	0	1	FALSE	1	chr: 360
os	0	1	FALSE	5	Mac: 247, Win: 109, Lin: 2, Chr: 1
status	0	1	FALSE	1	suc: 360
term	0	1	FALSE	1	win: 360

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	min	median	max	hist
difficulty	0	1	3.12	0.97	1.00	3.0	5.00	
confidence	0	1	3.18	0.99	1.00	3.0	5.00	
enjoyment	0	1	3.23	1.13	1.00	3.0	5.00	
age	0	1	20.53	2.29	18.00	20.0	37.00	
width	0	1	1512.24	255.52	1134.00	1440.0	3840.00	
height	0	1	802.53	127.14	680.00	769.0	2036.00	
violations	0	1	1.35	0.58	1.00	1.0	3.00	
absolute_score	0	1	2.02	3.62	0.00	0.0	12.00	
totaltime_m	0	1	13.00	14.83	2.82	11.3	253.84	

```
codebook(df_participants, #ONLY FOR HTML KNIT
         metadata_table = TRUE,
         detailed_variables = FALSE,
         detailed_scales = FALSE,
         metadata_json = FALSE,
         survey_overview = FALSE,
         missingness_report = FALSE)
```

Items Codebook

```
#see https://cran.r-project.org/web/packages/codebook/vignettes/codebook_tutorial.html

#ADD VARIABLE METADATA
dict <- rio::import("input/dictionary_sgc4a_items.csv", "csv") #import data dictionary

var_label(df_items) <- dict %>% select(VARIABLE, DESCRIPTION) %>% dict_to_list() #add variable labels
```

```
#ADD DATASET METATDATA
metadata(df_items)$name <- "Experimental ITEMS for study SGC4A"
metadata(df_items)$description <- "Data for study SGC4A summarized at participant-item level"
metadata(df_items)$creator <- "Amy Rae Fox"
metadata(df_items)$contact <- "amyraefox@gmail.com"
```

```
#{r, eval = checkMode() == "pdf"} #ONLY FOR PDF EXPORT
skim_codebook(df_items)
```

```
## Warning in sorted_count(x): Variable contains value(s) of "" that have been
## converted to "empty".
```

Table 5: Data summary

Name	data
Number of rows	5400
Number of columns	23
Column type frequency:	
character	4
factor	15
numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
study	0	1	5	5	0	1	0
graph	0	1	10	10	0	1	0
question	0	1	26	87	0	15	0
answer	0	1	0	27	138	176	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
subject	0	1	FALSE	360	075: 15, 07F: 15, 083: 15, 0BT: 15
term	0	1	FALSE	1	win: 5400
pool	0	1	FALSE	1	son: 5400
mode	0	1	FALSE	1	asy: 5400
condition	0	1	FALSE	4	115: 1470, 111: 1320, 114: 1320, 113: 1290
pretty_condition	0	1	FALSE	4	Ort: 1470, Ort: 1320, Ort: 1320, Tri: 1290
block	0	1	FALSE	2	ite: 4320, ite: 1080
explicit	0	1	FALSE	1	1: 5400
impasse	0	1	FALSE	1	1: 5400
grid	0	1	FALSE	4	5: 1470, 1: 1320, 4: 1320, 3: 1290
mark	0	1	FALSE	2	emp: 4080, 1: 1320
ixn	0	1	FALSE	1	1: 5400

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
relation	0	1	FALSE	10	end: 720, mee: 720, mid: 720, sta: 720
q	0	1	FALSE	15	1: 360, 2: 360, 3: 360, 4: 360
correct	0	1	FALSE	2	FAL: 3973, TRU: 1427

Variable type: numeric

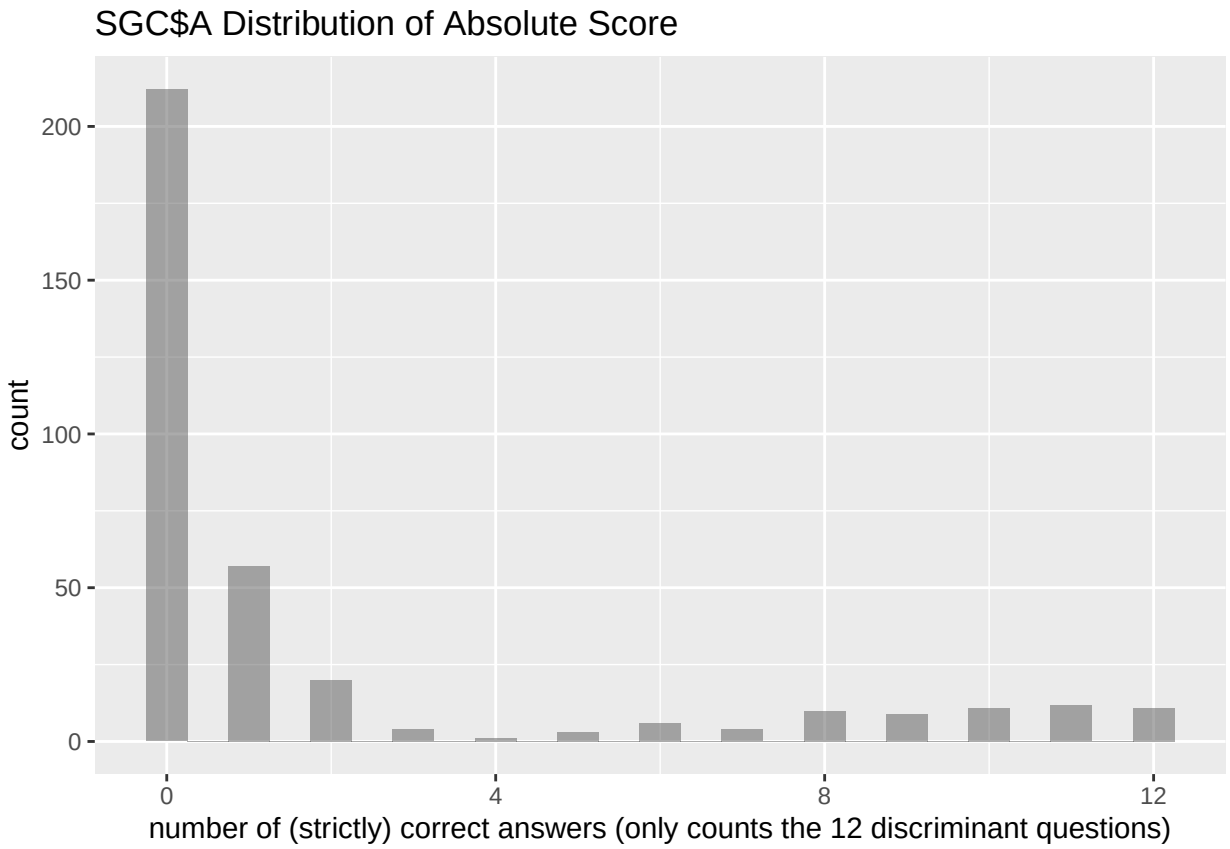
skim_variable	n_missing	complete_rate	mean	sd	min	median	max	hist
gwidth	0	1	600.00	0.00	600.00	600.00	600.00	▁▁▇▁▁
gheight	0	1	600.00	0.00	600.00	600.00	600.00	▁▁▇▁▁
time_elapsed_m	0	1	6.92	13.57	0.36	5.27	252.14	▇▁▁▁▁
rt_s	0	1	33.13	35.95	0.78	21.80	536.39	▇▁▁▁▁

```
codebook(df_items,#ONLY FOR HTML EXPORT
         metadata_table = TRUE,
         detailed_variables = FALSE,
         detailed_scales = FALSE,
         metadata_json = FALSE,
         survey_overview = FALSE,
         missingness_report = FALSE)
```

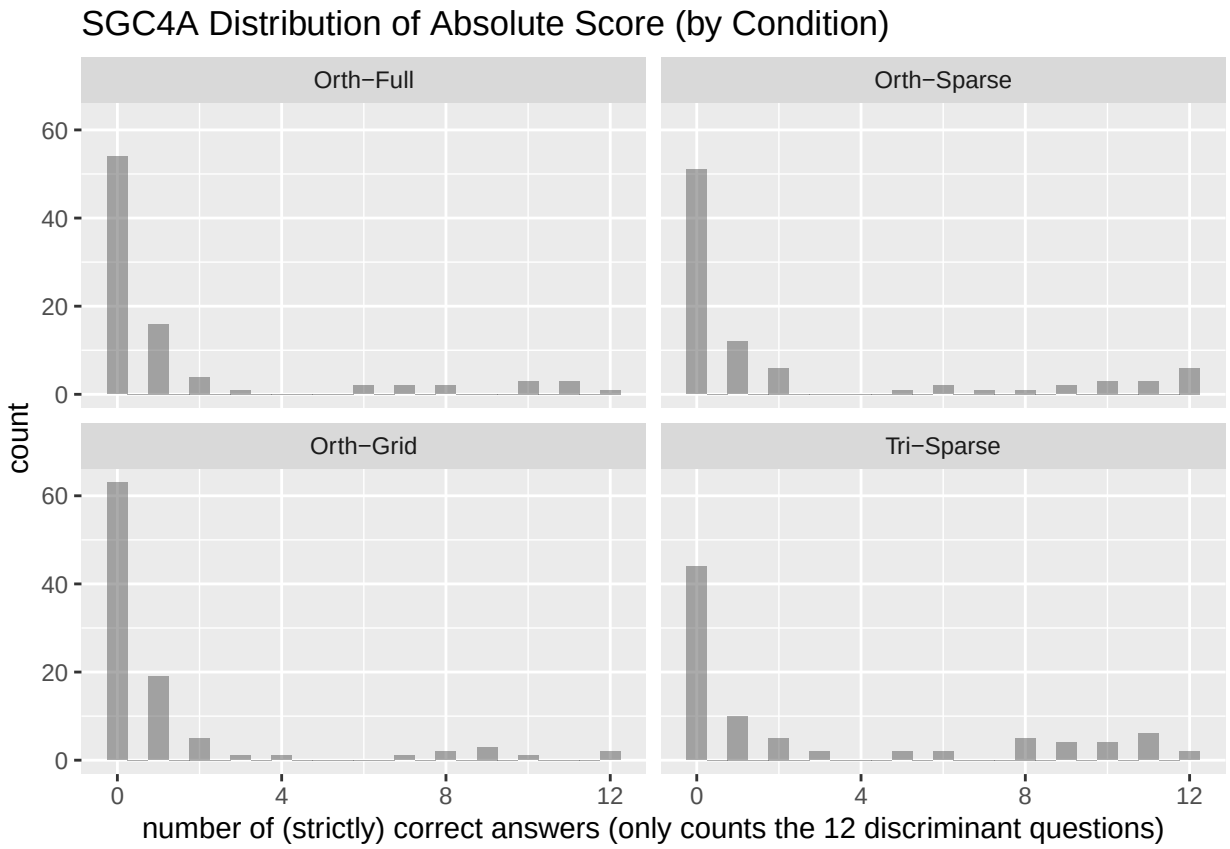
Explore

Exploration of the distribution of key response variables for validation purposes:

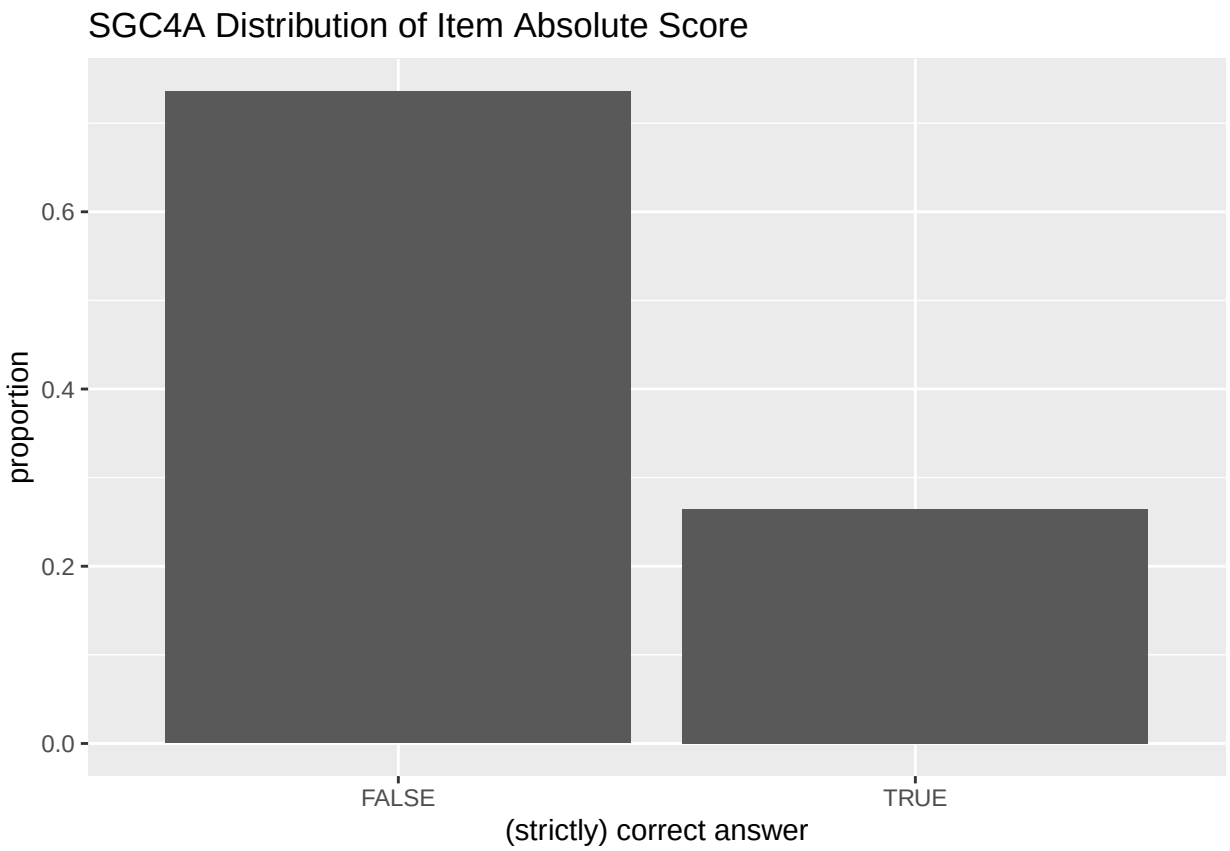
```
gf_histogram( ~absolute_score ,data = df_participants) +
  labs(title = "SGC$A Distribution of Absolute Score")
```



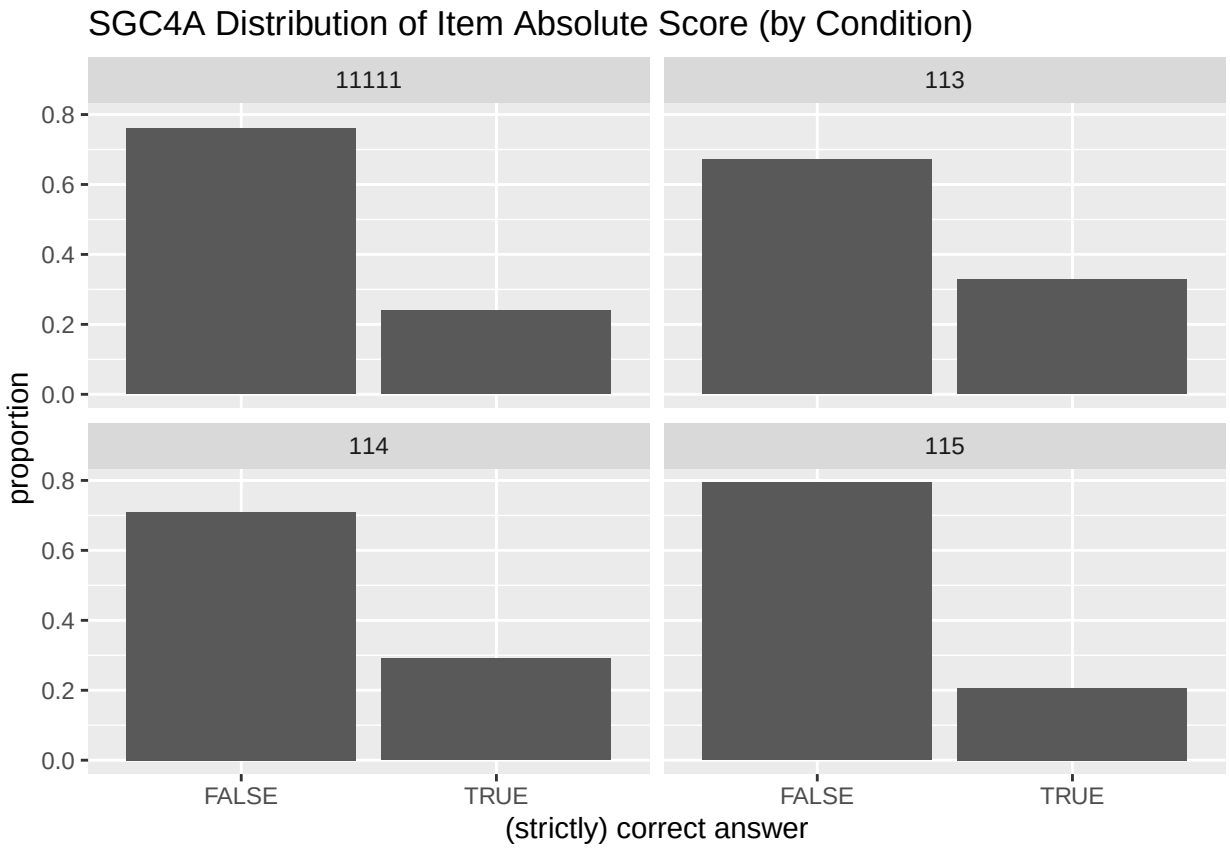

```
gf_histogram( ~absolute_score ,data = df_participants) %>%
  gf_facet_wrap(~pretty_condition) +
  labs(title = "SGC4A Distribution of Absolute Score (by Condition)")
```



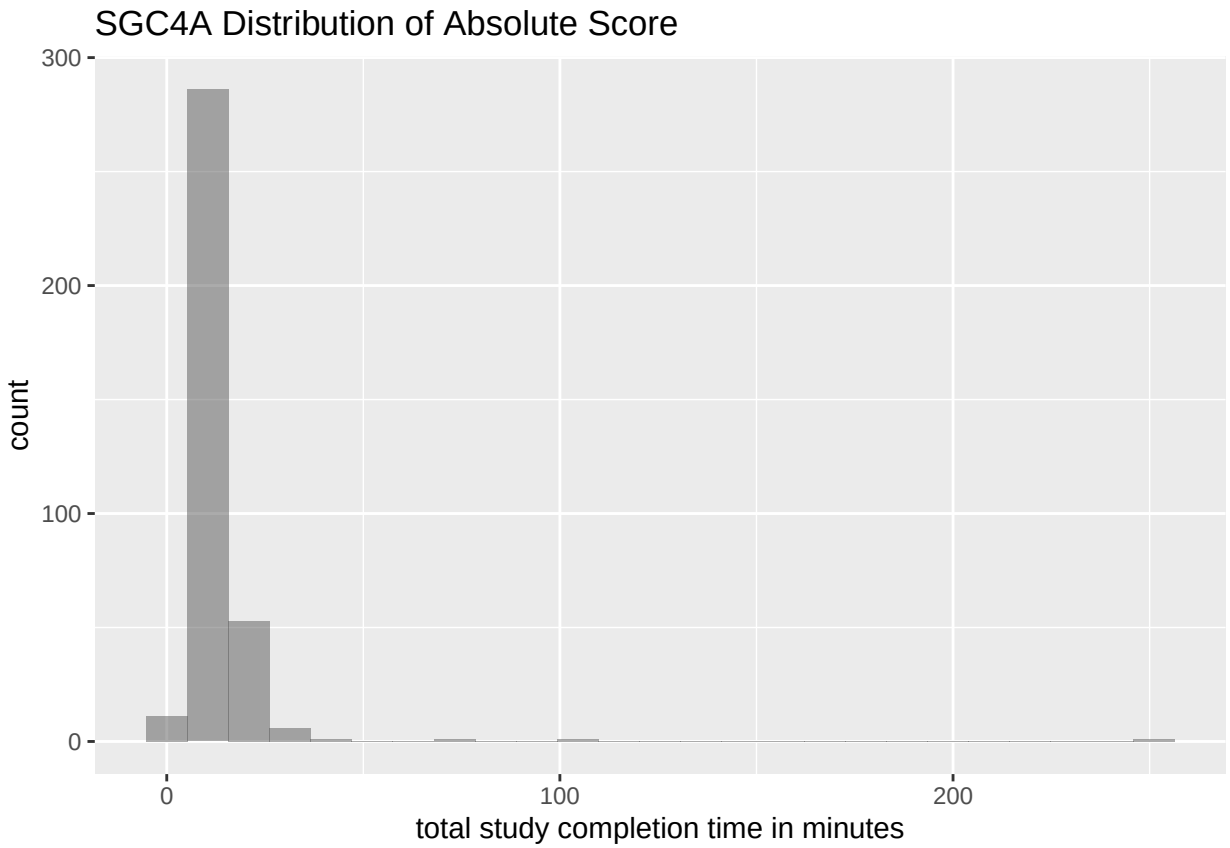

```
gf_props(~correct, data = df_items) +
  labs(title = "SGC4A Distribution of Item Absolute Score")
```



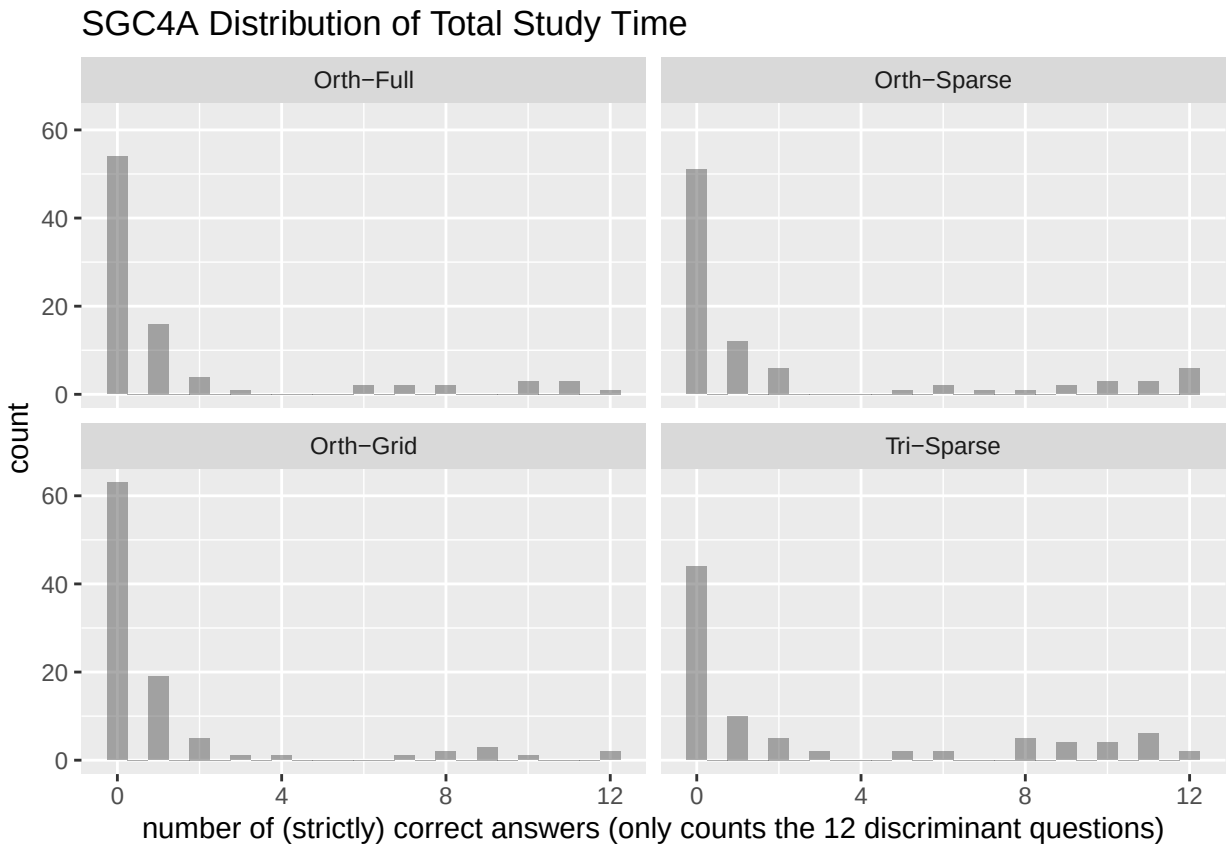

```
gf_props(~correct, data = df_items) %>%
  gf_facet_wrap(~condition) +
  labs(title = "SGC4A Distribution of Item Absolute Score (by Condition)")
```



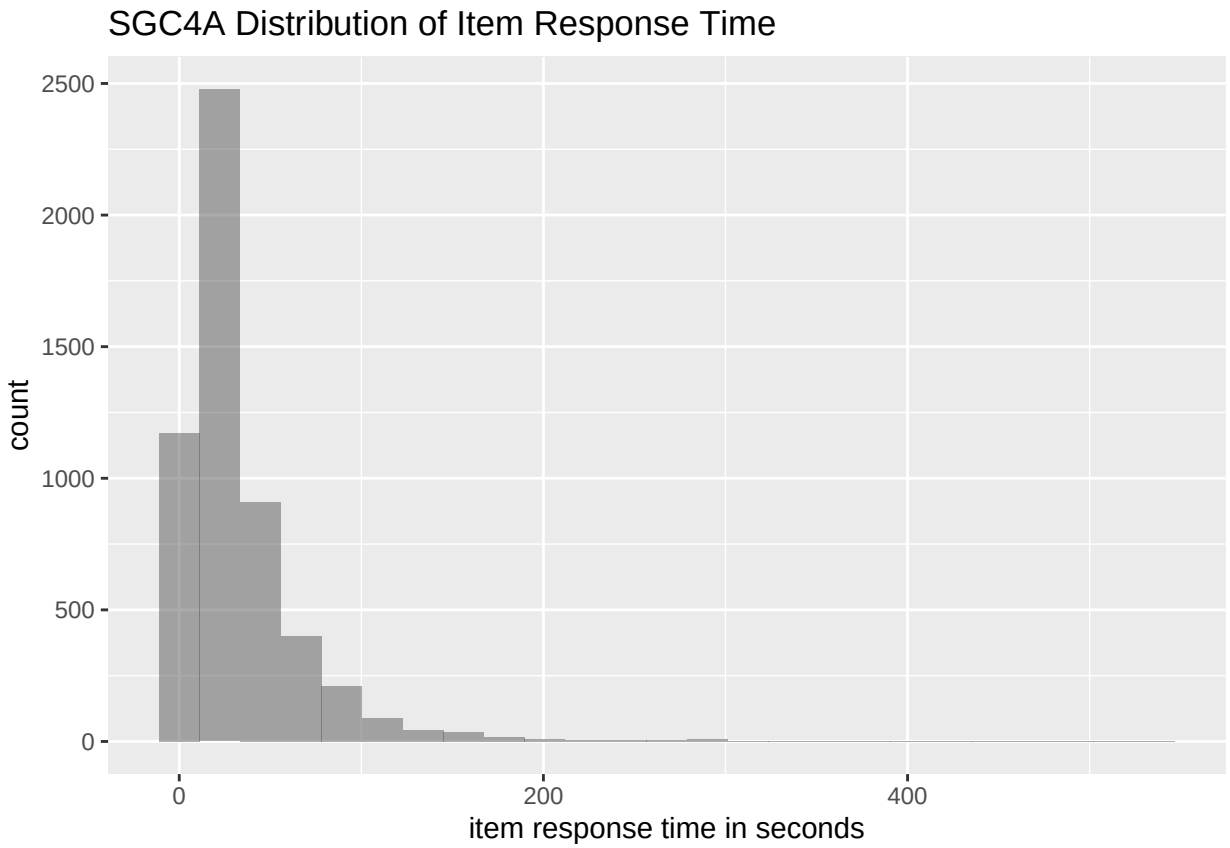

```
gf_histogram( ~totaltime_m ,data = df_participants) +
  labs(title = "SGC4A Distribution of Absolute Score")
```




```
gf_histogram( ~absolute_score ,data = df_participants) %>%
  gf_facet_wrap(~pretty_condition) +
  labs(title = "SGC4A Distribution of Total Study Time")
```




```
gf_histogram(~rt_s, data = df_items) +
  labs(title = "SGC4A Distribution of Item Response Time")
```



Data Export

Save Exclusions

For transparency, we save and identify the excluded data.

```
write.csv(ex_participants,"output/winter22_excluded_participants_sgc4A.csv", row.names = FALSE)
write.csv(ex_items,"output/winter22_excluded_items_sgc4A.csv", row.names = FALSE)
```

Analysis-Ready Files

Finally, we generate analysis ready files as .csv and .rds(containing data dictionary metatdata)

```
#save participant file
write.csv(df_participants,"output/winter22_sgc4a_participants.csv", row.names = FALSE)
#save item file
write.csv(df_items,"output/winter22_sgc4a_items.csv", row.names = FALSE)

#export R DATA STRUCTURES (include codebook metadata)
rio::export(df_participants, "output/winter22_sgc4a_participants.rds") # to R data structure file
rio::export(df_items, "output/winter22_sgc4a_items.rds") # to R data structure file
```