SGC3A

Study SGC3A | 3 Exploration

Amy Rae Fox

4/29/2022

Table of contents

Status

- 5/10/22 | SGC3A SCORING | calculate tversky subscores, start derive interpretation from subscores, refactor scoring cell as functions
- 5/10/22 | SGC3A SCORING | calculate nice_ABS (dichotomous score ignoring 'allowed' false-correct options (such as reference point)
- 5/5/22 - 5/8/22 | SGC3A SCORING | explore specific responses on each item
- 5/4/22 | SGC3A SCORING | replaced scoring strategy partial[-1/n, +1/n] to partial [-1/q, + 1/p]
- 4/29/22 | ported existing .Rmd analysis files to Quarto (.qmd) for sharing status w/ JMH CMW via web

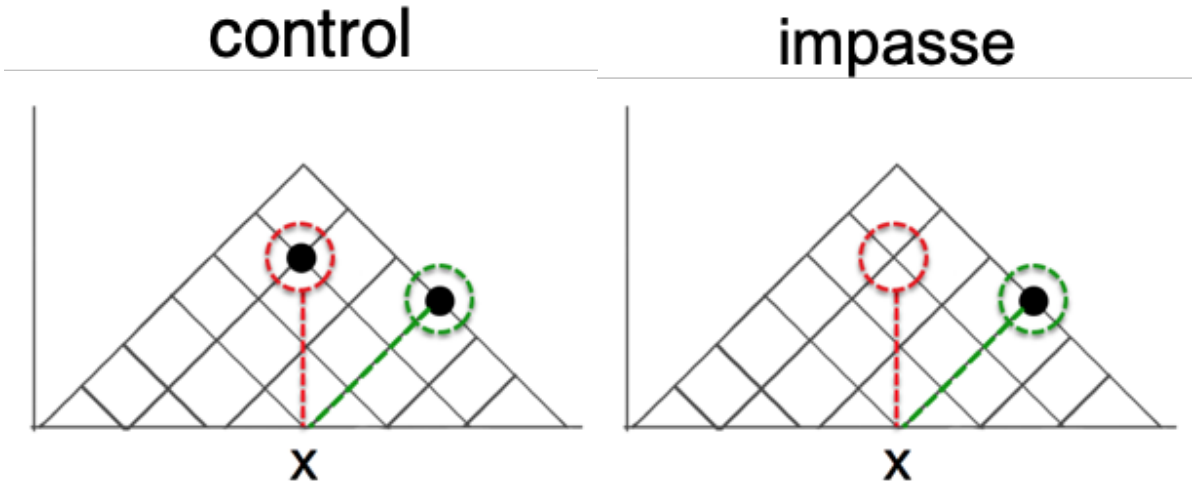
Part I

SGC3A

INTRODUCTION

In Study 3A we explore a hypothesis that emerged from analysis of Study 2, namely that **presenting a learning with a situation that induces a state of impasse will increase the probability that learners experience a moment of insight, and in turn restructure their interpretation of the coordinate system.**

In the context of Study 2, an impasse state was (unintentionally) induced when the combination of question + data set yielded no available answer in the incorrect (cartesian) interpretation of the graph. In Study 3A, we test this hypothesis by comparing performance between a (treatment) group receiving impasse-inducing questions followed by normal questions, and a non-impasse control.



(a) **?(caption)** (b) **?(caption)**

Figure 1: Posing a mental impasse

To try the study yourself:

- control condition
- impasse condition

Hypotheses

H1. Learners posed with impasse-inducing questions will be *more likely* to correct interpret the graph.

H0. Learners posed with impasse-inducing questions will be *no more likely* to correctly interpret the graph.

METHODS

Design

We employed a mixed design with 1 between-subjects factor with 2 levels (Scaffold: control, impasse) and 15 items (within-subjects factor).

Independent Variables:

- B-S (Scaffold: control,impasse)
- W-S (Item x 15)

Dependent Variables:

- Response Latency : Time from stimulus onset to clicking 'Submit' button: time in (s)
- Response Accuracy : Is the response triangular-correct?
- (derived) Interpretation : With which interpretation of the graph is the subject's response on an individual question consistent?

Materials

Stimuli consisted of a series of 15 graph comprehension questions, each testing a different combination of time interval relations, to be read from a Triangular-Model graph. Figure 2. The list of questions can be found here.

Note that across both control and impasse conditions, both the question, response options and graph structure were identical. The experimental manipulation (posing a mental impasse) was accomplished by changing the position of datapoints in the impasse-condition graph, such that for any given question, there was no available response option if the reader were to interpret the graph as cartesian (making an orthogonal rather than diagonal projection from the x-axis.)

The green line indicates the ideal-scanpath to the correct (triangular) answer to the first question, and the red line indicates the (incorrect) orthogonal interpretation. In the IMPASSE figure (at right), there are no data points that intersect the red line.

Procedure

Participants completed the study via a web-browser. Upon starting, they submitted informed consent, before reading task instructions. Participants were introduced to a scenario in which they were to play the role of a project manager, scheduling shifts for a group of employees. The schedule of the employees was presented in a TriangularModel (TM) graph, and they would be answering question about the schedule. Then participants completed a test block of 15 items. In the IMPASSE condition, the first five questions included an IMPASSE problem

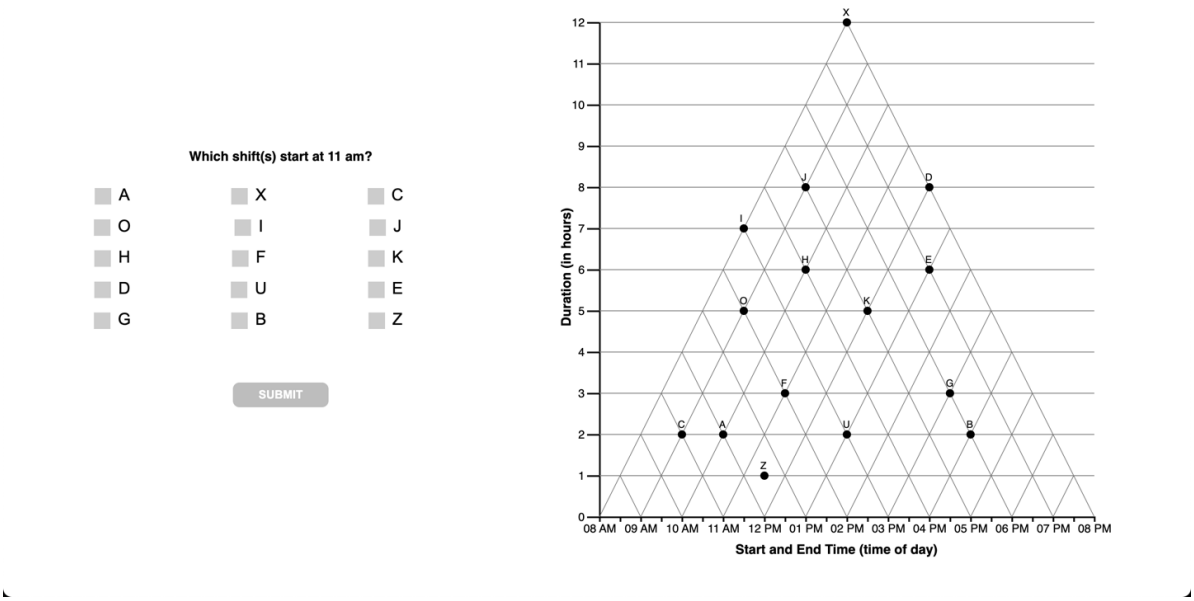


Figure 2: Sample Question (Q=1) for Graph Comprehension Task

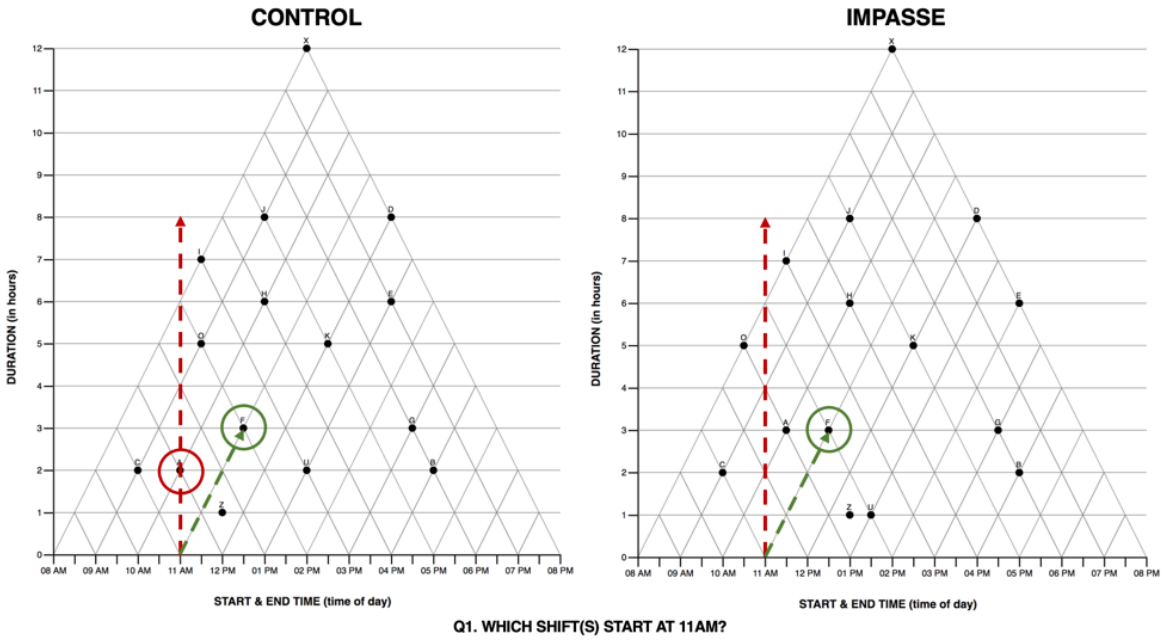


Figure 3: Sample Question (Q=1) graphs for each condition

state. For participants in the CONTROL condition, the dataset was structure such that there was always an available 'orthogonal answer' for the first 5 questions. In both conditions, the remaining 10 questions were not structured as impasse. Following the test block, participants answered a free-response question about their strategy for reading the graph, followed by a demographic questionnaire and debrief.

Sample

Data was collected by convenience sample of a university subject pool. Initial data (Fall 2017, Spring 2018) were collected in-person, with large groups of students simultaneously completing the study (independently) in a computer lab. In Fall 2021 and Winter 2022 we collected additional data to replicate results in a remote format (students completing the study asynchronously on their own computers).

ANALYSIS

Data Preparation

Before our data can be analyzed, data files from individual data collection periods must be harmonized into a common data format (Section 1).

Response Scoring

Because the graph comprehension task utilizes a Multiple-Response (MR) format (rather than simple multiple choice), the raw response data (the combination of answer options selected) for each question first need to be assigned a score. Approaches to scoring MR data and score transformations are derived in Section 2.

1 Harmonization

The purpose of this notebook is to harmonize data files for study SGC_3A.

Pre-Requisite	Followed By
spring17_clean_data.Rmd spring18_clean_data.Rmd fall21_clean_data.Rmd winter2022_clean_sgc3a.Rmd	2_sgc3A_rescoring.qmd

1.1 HARMONIZATION

Data for study SGC_3A were collected across four time periods, interrupted by the Covid-19 pandemic.

Period	Modality
Fall 2017	in person, SONA groups in computer lab
Spring 2018	in person, SONA groups in computer lab
Fall 2021	asynchronous, online, SONA
Winter 2022	asynchronous, online, SONA

Data collected in Fall 2017, Spring 2018 constitute the original SGC_3A study, conducted in person. Data collected in Fall 2021, Winter 2022 constitute the web-based replication, conducted online (asynchronously). In all cases, the experiment was administered via a web application.

The underlying data structure of the stimulus web application changed across the data collection period, resulting in slightly different data files (i.e. columns are not named consistently). In this section, we combine the files from each data collection period into a single *harmonized* data file for analysis (one for participants, one for items).

1.1.1 Participants

First we import participant-level data from each data collection period, selecting only the columns relevant for analysis, and renaming columns to be consistent across each file. The result is a single data frame `df_subjects` containing one row for each subject (across all periods). Note that we *are not* discarding any *response* data. Rather, we discard columns that are automatically recorded by the stimulus web application and help the application run.

Note that we discard some columns representing scores calculated in the stimulus engine. These scores were calculated differently across collection periods, and so we discard them and recalculate scores in the next analysis notebook.

```
#IMPORT PARTICIPANT DATA

#set datafiles
fall17 <- "data/session-level/fall17_sgc3a_participants.csv"
spring18 <- "data/session-level/spring18_sgc3a_participants.csv"
fall21 <- "data/session-level/fall21_sgc3a_participants.csv"
winter22 <- "data/session-level/winter22_sgc3a_participants.rds"

#read datafiles, set mode and term
df_subjects_fall17 <- read.csv(fall17) %>% mutate(mode = "lab-synch", term = "fall17")
df_subjects_spring18 <- read.csv(spring18) %>% mutate(mode = "lab-synch", term = "spring18
df_subjects_fall21 <- read.csv(fall21) %>% mutate(mode = "asynch", term = "fall21")
df_subjects_winter22 <- read_rds(winter22) #use RDS file as it contains metadata

#SAVE METADATA FROM WINTER, but no rows
df_subjects <- df_subjects_winter22 %>% filter(condition=='X') %>% select(
  subject,condition,term,mode,
  gender,age,language, schoolyear, country,
  effort,difficulty,confidence,enjoyment,other,
  totaltime_m,absolute_score
)

#reduce data collected using OLD webapp to useful columns
df_subjects_before <- rbind(df_subjects_fall17, df_subjects_spring18, df_subjects_fall21)
  #rename and summarize some columns
  mutate(
    totaltime_m = totalTime / 1000 / 60,
    absolute_score = triangular_score,
    language = native_language,
    gender = sex,
```

```
    schoolyear = year) %>%
  #create placeholders for cols not collected until NEW webapp [for later rbind]
  mutate(
    effort = "NULL",
    difficulty = "NULL",
    confidence = "NULL",
    enjoyment = "NULL",
    other = "NULL",
    disability = "NULL"
  ) %>%
  #select only columns we'll be analyzing, discard others
  dplyr::select(subject, condition, term, mode,
                #demographics
                gender, age, language, schoolyear, country,
                #placeholder effort survey
                effort, difficulty, confidence, enjoyment,
                #placeholder misc
                other, disability,
                #response characteristics
                totaltime_m, absolute_score)

#save 'explanation' columns from winter22, which is actually a response to a free response
df_winter22_q16 <- df_subjects_winter22 %>%
  select(subject, condition, term , mode, explanation) %>%
  mutate(
    q = 16,
    response = explanation
  ) %>% select(-explanation)

#reduce data collected using NEW webapp to useful columns
df_subjects_winter22 <- df_subjects_winter22 %>%
  mutate(score = absolute_score) %>%
  #select only columns we'll be analyzing, discard others
  dplyr::select( subject, condition, term, mode,
                 #demographics
                 gender, age, language, schoolyear, country,
                 #effort survey
                 effort, difficulty, confidence, enjoyment,
                 #explanations
                 other,disability,
                 #response characteristics
```

```
                totaltime_m, absolute_score)


effort_labels <- c("I tried my best on each question", "I tried my best on most questions"

#combine dataframes from old and new webapps
df_subjects <- rbind(df_subjects, df_subjects_winter22, df_subjects_before) %>%
  #refactor factors
  mutate (
    subject = factor(subject),
    condition = factor(condition),
    term = factor(term),
    mode = factor(mode),
    gender = factor(gender),
    schoolyear = as.factor(schoolyear)
  )

#FIX METADATA
#Add metadata for columns that lost it [factors, for some reason!]
var_label(df_subjects$subject) <- "ID of subject (randomly assigned in stimulus app)."
var_label(df_subjects$condition) <- "ID indicates randomly assigned condition (111 -> cont
var_label(df_subjects$term) <- "indicates if session was run with experimenter present or
var_label(df_subjects$mode) <- "indicates mode in which the participant completed the stud
var_label(df_subjects$gender) <- "What is your gender identity?"
var_label(df_subjects$schoolyear) <- "What is your year in school?"

#CLEANUP
rm(df_subjects_fall17,df_subjects_fall21, df_subjects_spring18, df_subjects_winter22,df_su
rm(fall17,fall21,spring18,winter22)
```

1.1.2 Items

Next we import item-level data from each data collection period, selecting only the columns relevant for analysis, and renaming columns to be consistent across each file. The result is a single data frame `df_items` containing one row for each *graph comprehension task question* (qs=15) (across all periods). A second data frame `df_freeresponse` contains one row for each free response strategy question (last question posed to participants in Winter2022) Note that we *do not* discard any *response* data. Rather, we *do* discard several columns representing accuracy scores for responses that were calculated in the stimulus engine. These scores were calculated differently across collection periods, and so we discard them and recalculate scores in the next analysis notebook. Original response data are always preserved.

```r
#set datafiles
fall17 <- "data/session-level/fall17_sgc3a_blocks.csv"
spring18 <- "data/session-level/spring18_sgc3a_blocks.csv"
fall21 <- "data/session-level/fall21_sgc3a_blocks.csv"
winter22 <- "data/session-level/winter22_sgc3a_items.rds"

#read datafiles, set mode and term
df_items_fall17 <- read.csv(fall17) %>% mutate(mode = "lab-synch", term = "fall17")
df_items_spring18 <- read.csv(spring18) %>% mutate(mode = "lab-synch", term = "spring18")
df_items_fall21 <- read.csv(fall21) %>% mutate(mode = "asynch", term = "fall21")
df_items_winter22 <- read_rds(winter22) #use RDS file as it contains metadata

#get mapping being question # and interval relation the question tests, that is encoded on
map_relations <- df_items_winter22 %>% group_by(q) %>% select(q,relation) %>% unique()


#SAVE METADATA FROM WINTER, but no rows
df_items <- df_items_winter22 %>% filter(condition=='X') %>% select(
  subject,condition,term,mode,
  question, q, answer, correct, rt_s
)

#reduce data collected using old webapp
df_items_before <- rbind(df_items_fall17, df_items_spring18, df_items_fall21) %>%
  mutate(rt_s = rt / 1000, correct = as.logical(correct)) %>%
  select(subject, condition, term, mode, question, q, answer, correct, rt_s)

#reduce data collected using new webapp
df_items_winter22 <- df_items_winter22 %>%
  select(subject, condition, term, mode, question, q, answer, correct, rt_s) %>% #unfactor
  mutate(
    subject = as.character(subject),
    condition = as.character(condition),
    term = as.character(term),
    mode = as.character(mode),
    q = as.integer(q),
    correct = as.logical(correct)
  )

#combine dataframes from old and new webapps
df_items <- rbind(df_items, df_items_winter22,df_items_before) %>%
  #refactorize columns
```

```
  mutate(
    subject = factor(subject),
    condition = factor(condition),
    term = factor(term),
    mode = factor(mode),
    q = as.integer(q)) %>%
  #rename answer column to RESPONSE
  rename(response = answer) %>%
  #remove all commas and make as character string
  mutate(
    response = str_remove_all(as.character(response), ","),
    num_o = str_length(response)
  )


#FIX METADATA
#Add metadata for columns that lost it [factors, for some reason!]
var_label(df_items$subject) <- "ID of subject (randomly assigned in stimulus app)."
var_label(df_items$condition) <- "ID indicates randomly assigned condition (111 -> control
var_label(df_items$term) <- "indicates if session was run with experimenter present or asy
var_label(df_items$mode) <- "indicates mode in which the participant completed the study"
var_label(df_items$q) <- "Question Number (in order)"
var_label(df_items$correct) <- "Is the response (strictly) correct? [dichotomous scoring]"
var_label(df_items$response) <- "options (datapoints) selected by the subject"
var_label(df_items$num_o) <- "number of options selected by the subject"

#HANDLE FREE RESPONSE QUESTION #16
#save `free response` Q#16 in its own dataframe
df_freeresponse <- df_items %>% filter(q == 16) %>% select(-question,-correct,-rt_s,-num_o
#add data from wi22 [stored on subject data]
df_freeresponse <- rbind(df_freeresponse, df_winter22_q16)
#add question description
df_freeresponse <- df_freeresponse %>% mutate(
    question = "Please describe how to determine what event(s) start at 12pm?",
    response = as.character(response) #doesn't need to be factor
  )
#remove 'free response' Q#16 from df_items
df_items <- df_items %>% filter (q != 16)

#CLEANUP
rm(df_items_fall17,df_items_fall21, df_items_spring18, df_items_winter22, df_items_before,
```

```
rm(fall17,fall21,spring18,winter22, map_relations)
```

1.1.2.1 Validation

Next, we validate that we have the complete number of item-level records based on the number of subject-level records

```
#the number of items should be equal to 15 x the number of subjects
nrow(df_items) == 15* nrow(df_subjects) #TRUE
```

```
[1] TRUE
```

```
#each subject should have 15 items
df_items %>% group_by(subject) %>% summarise(n = n()) %>% filter(n != 15) %>% nrow() == 0
```

```
[1] TRUE
```

1.2 EXPORT

Finally, we export the (session-harmonized) data for analysis, as CSVs, and .RDS (includes metadata)

```
#SAVE FILES
write.csv(df_subjects,"data/sgc3a_participants.csv", row.names = FALSE)
write.csv(df_items,"data/sgc3a_items.csv", row.names = FALSE)
write.csv(df_freeresponse,"data/sgc3a_items.csv", row.names = FALSE)

#SAVE R Data Structures
#export R DATA STRUCTURES (include codebook metadata)
rio::export(df_subjects, "data/sgc3a_participants.rds") # to R data structure file
rio::export(df_items, "data/sgc3a_items.rds") # to R data structure file
```

1.3 RESOURCES

```
sessionInfo()
```

```
R version 4.0.2 (2020-06-22)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS  10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] codebook_0.9.2  forcats_0.5.0   stringr_1.4.0   dplyr_1.0.2
 [5] purrr_0.3.4     readr_1.4.0     tidyr_1.1.2     tibble_3.1.2
 [9] ggplot2_3.3.5   tidyverse_1.3.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.5         lubridate_1.7.9    assertthat_0.2.1  digest_0.6.27
 [5] utf8_1.2.1         R6_2.5.0           cellranger_1.1.0  backports_1.2.1
 [9] reprex_0.3.0       labelled_2.8.0     evaluate_0.14     httr_1.4.2
[13] pillar_1.6.1       rlang_0.4.11       curl_4.3          readxl_1.3.1
[17] rstudioapi_0.13    data.table_1.13.2  blob_1.2.1        rmarkdown_2.11
[21] foreign_0.8-80     munsell_0.5.0      broom_0.7.12      compiler_4.0.2
[25] modelr_0.1.8       xfun_0.29          pkgconfig_2.0.3   htmltools_0.5.2
[29] tidyselect_1.1.0   rio_0.5.16         fansi_0.5.0       crayon_1.4.1
[33] dbplyr_1.4.4       withr_2.4.2        grid_4.0.2        jsonlite_1.7.1
[37] gtable_0.3.0       lifecycle_1.0.0    DBI_1.1.0         magrittr_2.0.1
[41] scales_1.1.1       zip_2.1.1          cli_3.3.0         stringi_1.7.3
[45] fs_1.5.0           xml2_1.3.2         ellipsis_0.3.2    generics_0.0.2
[49] vctrs_0.3.8        openxlsx_4.2.3     tools_4.0.2       glue_1.6.2
[53] hms_0.5.3          fastmap_1.1.0      yaml_2.2.1        colorspace_2.0-2
[57] rvest_0.3.6        knitr_1.37         haven_2.3.1
```

2 Response Scoring

```
# knitr::opts_chunk$set(eval = FALSE) #knit w/o execution
options(scipen=1, digits=3)

library(tidyverse) #ALL THE THINGS
library(kableExtra) #printing tables
library(huxtable) #tables
library(ggformula) #quick graphs
```

THIS NOTEBOOK IS INCOMPLETE

The purpose of this notebook is to score (assign a measure of accuracy) to response data for the SGC_3A study. This is required because the question type on the graph comprehension task used a 'Multiple Response' (MR) question design. Here, we evaluate different approaches to scoring multiple response questions, and transform raw item responses (e.g. boxes ABC are checked) to a measure of response accuracy. (Warning: this notebook takes several minutes to execute.)

Pre-Requisite	Followed By
1_sgc3a_harmonize.qmd	3_sgc3A_exploration.qmd

2.1 MULTIPLE RESPONSE SCORING

The *graph comprehension task* of study SGC 3A presents readers with a graph, a question, and a series of checkboxes. Participants are instructed to use the graph to answer the question, and respond by selecting all the checkboxes that apply, where each checkbox corresponds to a datapoint in the graph.

In the psychology and education literatures on Tests & Measures, the format of this type of question is referred to as Multiple Response (MR), (also: Multiple Choice Multiple Answer (MCMA) and Multiple Answer Multiple Choice (MAMC)). It has a number of properties that make it different from traditional Single Answer Multiple Choice (SAMC) questions, where the respondent marks a single response from a number of options. In particular, there are a number of very different ways that MAMC questions can be *scored*.

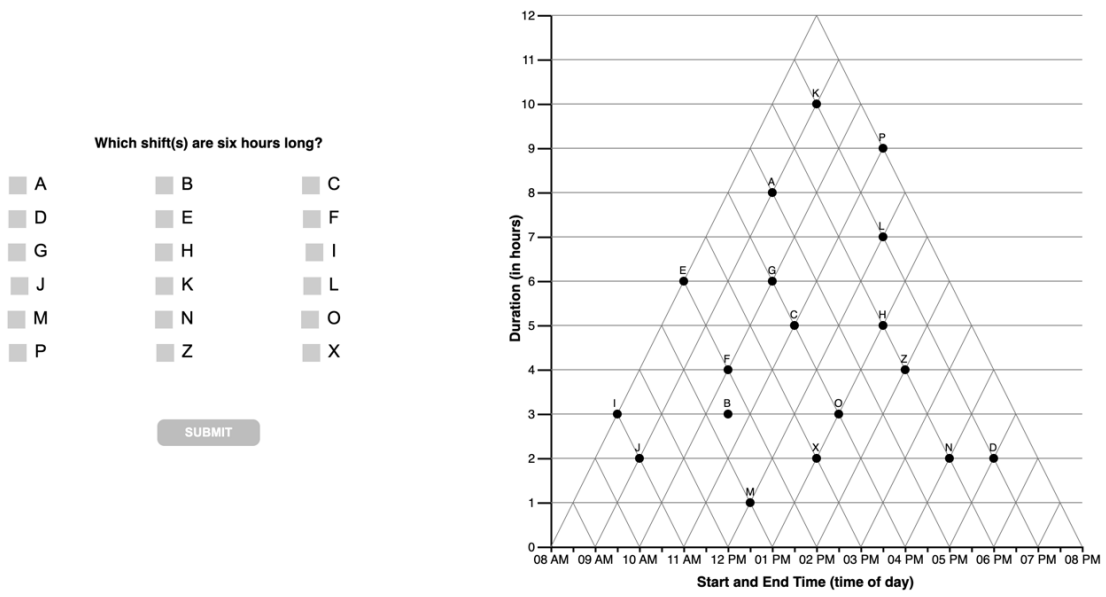


Figure 2.1: **Figure 1. Sample Graph Comprehension (Question # 6)**

In tranditional SAMC format questions, one point is given for selecting the option designated as correct, and zero points given for marking any of the alternative (i.e. distractor) options. Individual response options on MAMC questions, however might be partially correct (i), while responses on other answer options within the same item might be incorrect (n–i). In MR, it is not obvious how to allocate points when the respondent marks a true-correct option (i.e. options that *should* be selected, denoted p), as well as one or more false-correct options (i.e. options that *should not* be selected, denoted q). Should partial credit be awarded? If so, are options that respondents false-selected and false-unselected items equally penalized?

Schmidt et al. (2021) performed a systematic literature review of publications proposing MAMC (or equivalent) scoring schemes, ultimately synthesizing over 80 sources into 27 distinct scoring approaches. Upon reviewing the benefits of trade-offs of each approach, for this study we choose utilize two of the schemes: **dichotomous scoring** (Schmidt et al. (2021) scheme #1), and **partial scoring** $[-1/q, 0, +1/p]$ (Schmidt et al. (2021) scheme #26), as well as a scaled **discriminant score** that leverages partial scoring to discriminate between strategy-specific patterns of response.

2.1.1 Response Encoding

First, we note that the question type evaluated by Schmidt et al. (2021) is referred to as *Multiple True-False* (MTF), a variant of MAMC where respondents are presented with a question

(stem) and series of response options with True/False (e.g. radio buttons) for each. Depending on the implementation of the underlying instrument, it may or may not be possible for respondents to *not respond* to a particular option (i.e. leave the item 'blank'). Although MTF questions have a different underlying implementation (and potentially different psychometric properties) they are identical in their mathematical properties; that is, responses to a MAMC question of 'select all that apply' can be coded as a series of T/F responses to each response option

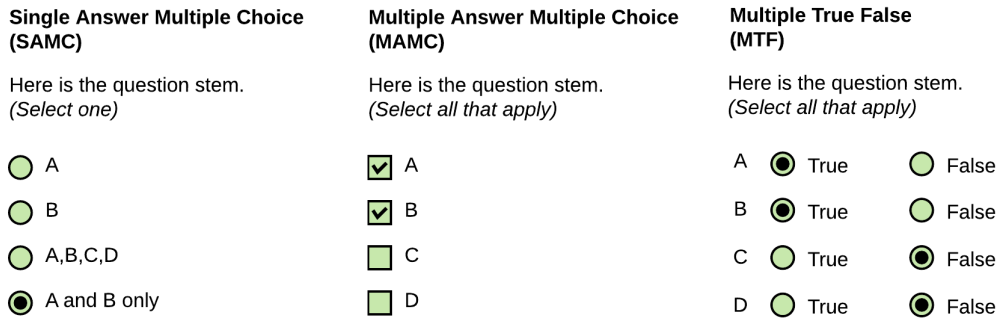


Figure 2.2: **Figure 2. SAMC (vs) MAMC (vs) MTF**

In this example (Figure 2.2), we see an example of a question with four response options ($n = 4$) in each question type. In the **SAMC** approach (at left), there are four possible responses, given explicitly by the response options (respondent can select only one) (number of possible responses $= n$). With only four possible responses, we cannot entirely discriminate between all combinations of the underlying response variants we might be interested in, and must always choose an 'ideal subset' of possible distractors to present as response options. In the MAMC (middle) and MTF (at right), the *same number of response options* ($n = 4$) yield a much greater number (number of possible responses $= 2^n$). We can also see the equivalence between a MAMC and MTF format questions with the same response options. Options the respondent *selects* in MAMC are can be coded as T, and options they leave *unselected* can be coded as F. Thus, for response options (ABCD), a response of [AB] can also be encoded as [TTFF].

2.1.2 Scoring Schemes

In the sections that follow, we use the terminology:

Properties of the Stimulus-Question

$$n = \text{number of response options} \tag{2.1}$$
$$= p + q \tag{2.2}$$
$$p = \text{number of true-select options (i.e. should be selected)} \tag{2.3}$$
$$q = \text{number of true-unselect options (i.e. should not be selected)} \tag{2.4}$$

Properties of the Subject's Response

$$i = \text{number of options in correct state}, (0 \leq i \leq n) \tag{2.5}$$
$$f = \text{resulting score} \tag{2.6}$$

2.1.2.1 Dichotomous Scoring

Dichotomous Scoring is the strictest scoring scheme, where a response only receives points if it is *exactly* correct, meaning the respondent includes *only correct-select* options, and does select any additional (i.e. incorrect-select) options that should not be selected. This is also known as *all or nothing scoring*, and importantly, it ignores any partial knowledge that a participant may be expressing through their choice of options. They may select some but not all of the correct-select options, and one or more but not all of the correct-unselect items, but receive the same score as a respondent selects none of the correct-select options, or all of the correct-unselect options. In this sense, dichotomous scoring tells us *only* about perfect knowledge, and ignores any indication of partial knowledge the respondent may be indicating through their selection of response options.

In Dichotomous Scoring

- score for the question is either 0 or 1
- full credit is only given if all responses are correct; otherwise no credit
- does not account for *partial knowledge*. - with increasing number of response options, scoring becomes stricter as each statement must be marked correctly.

The algorithm for **dichotomous scoring** is given by:

$$f = \begin{cases} 1, \text{if } i = n \\ 0, \text{otherwise} \end{cases}$$

where 0 i n

```
f_dichom <- function(i, n) {

  # print(paste("i is :",i," n is:",n))

  #if (n == 0 ) return error
  ifelse( (n == 0), print("ERROR n can't be 0"), "")

  #if (i > n ) return error
  ifelse( (i > n), print("i n can't > n"), "")

  #if (i==n) return 1, else 0
  return (ifelse( (i==n), 1 , 0))

}
```

2.1.2.2 Partial Scoring [-1/n, +1/n]

Partial Scoring refers to a class or scoring schemes that award the respondent partial credit depending on pattern of options they select. Schmidt et al. (2021) identify twenty-six different partial credit scoring schemes in the literature, varying in the range of possible scores, and the relative weighting of incorrectly selected (vs) incorrectly unselected options.

A particularly elegant approach to partial scoring is referred to as the $[-1/n, +1/n]$ approach (Schmidt et al. (2021) #17). This approach is appealing in the context of SGC3A, because it: (1) takes into account all information provided by the respondent: the pattern of what the select, and choose not to select.

In Partial Scoring $[-1/n, +1/n]$:

- Scores range from [-1, +1]
- One point is awarded if all options are *correct*
- One point point is subtracted if all options are *incorrect*.
- Intermediate results are credited as fractions accordingly ($+1/n$ for each correct, $-1/n$ for each incorrect)
- This results in *at chance performance* (i.e. half of the given options marked correctly), being awarded 0 points are awarded

This scoring is more consistent with the motivating theory that Triangular Graph readers start out with an incorrect (i.e. orthogonal, cartesian) interpretation of the coordinate system, and transition to a correct (i.e. triangular) interpretation. But the first step in making this transition is realizing the cartesian interpretation *is incorrect*, which may yield blank responses where the respondent is essentially saying, 'there is no correct answer to this question'.

Schmidt et al. (2021) describe the *Partial* $[-1/n, +1/n]$ scoring scheme as the *only* scoring method (of the 27 described) where respondents' scoring results can be interpreted as a percentage of their true knowledge. One important drawback of this method is that a respondent may receive credit (a great deal of credit, depending on the number of answer options n) even if she did not select *any* options. In the case (such as ours) where there are many more response options n than there are options meant to be selected p, this partial scoring algorithm poses a challenge because the respondent can achieve an almost completely perfect score by selecting a small number of options that should not be selected.

The algorithm for **partial scoring**$[-1/n, +1/n]$ is given by:

$$f = (1/n * i) - (1/n * (n - i)) \tag{2.7}$$
$$= (2i - n)/n \tag{2.8}$$

```
f_partialN <- function(i, n) {

# print(paste("i is :",i," n is:",n))

#if(n==0) return error
ifelse((n==0),print("ERROR: n should not be 0"),"")

#if(i >n ) return error
ifelse((i > n),print("ERROR: i CANNOT BE GREATER THAN n"),"")

return ((2*i - n) / n)
}
```

2.1.2.3 Partial Scoring [-1/p, +1/p]

One drawback of the Partial Scoring $[-1/n, +1/n]$ approach is that treats the choice to select, and choice to not select options as equally indicative of the respondent's understanding. That is to say, incorrectly selecting one particular option is no more or less informative than incorrectly not-selecting a different item. This represents an important difference between MAMC (i.e. "select all correct options") vs MTF (i.e. "Mark each option as true or false") questions.

In our study, the selection of any particular option (remember options represent data points on the stimulus graph) is indicative of a particular interpretation of the stimulus. Incorrectly selecting an option indicates an interpretation of the graph with respect to that particular option. Alternatively, failing to select a correct option *might* mean the individual has a different

interpretation, or that they failed to find *all* the data points consistent with the interpretation.

For this reason, we consider another alternative Partial Scoring scheme that takes into consideration only the selected statements, without penalizing statements incorrectly *not selected*. (See Schmidt et al. (2021) method #26; also referred to as the Morgan-Method) This partial scoring scheme takes into consideration that the most effort-free (or 'default') response for any given item is the null, or blank response. Blank responses indicate *no understanding*, perhaps *confusion*, or refusal to answer. These lack of responses are awarded zero credit. Whereas taking the action to select an *incorrect* option is effortful, and is indicative of *incorrect understanding*.

Partial Scoring $[-1/q, +1/p]$:

- awards +1/p for each correctly selected option (p_s), and subtracts $1/(n-p) = 1/q$ for each incorrectly selected option (q_s)
- only considers *selected* options; does not penalize nor reward *unselected* options

Properties of Item

$$p = \text{number of true-select options (i.e. should be selected)} \tag{2.9}$$
$$q = \text{number of true-unselect options (i.e. should not be selected)} \tag{2.10}$$
$$n = \text{number of options } (n = p + q) \tag{2.11}$$

Properties of Response

$$p_s = \text{number of true-select options selected (i.e. number of correctly checked options)} \tag{2.12}$$
$$q_s = \text{number of true-unselect options selected (i.e. number of incorrectly checked options} \tag{2.13}$$

The algorithm for **partial scoring** $[-1/q, +1/p]$ is given by:

$$f = (p_m/p) - (q_m/q) \tag{2.14}$$
$$\tag{2.15}$$

```
f_partialP <- function(t,p,f,q) {
```

```
  #t = number of correct-selected options
  #p = number of true options
  #f = number of incorrect-selected options
  #q = number of false options
  #n = number of options + p + q
  ifelse( (p == 0), return(""), "") #handle empty response set gracefully by returning not
  ifelse( (p != 0), return( (t / p) - (f/q)), "")

  # return( (t / p) - (f/q))
}
```

2.1.3 Comparison of Schemes

Which scoring scheme is most appropriate for the goals of the graph comprehension task?

Consider the following example:

For a question with $n = 5$ response options (data points A, B, C, D and E) with a correct response of A, the schemes under consideration yield the following scores:

```
title <- "Comparison of Scoring Schemes for n = 5 options [ A,B,C,D,E ]"

correct <- c( "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____" )

response <- c("A____",
              "AB___",
              "A___E",
              "AB__E",
              "____E",
              "___DE",
              "_BCDE",
              "ABCDE",
              "_____" )
```

```
i <- c(         5,
                4,
                4,
                3,

                3,
                2,
                0,
                1,
                4)

abs <- c(f_dichom(5,5),
         f_dichom(4,5),
         f_dichom(4,5),
         f_dichom(3,5),

         f_dichom(3,5),
         f_dichom(2,5),
         f_dichom(0,5),
         f_dichom(1,5),
         f_dichom(4,5))

partial1 <- c(f_partialN(5,5),
              f_partialN(4,5),
              f_partialN(4,5),
              f_partialN(3,5),

              f_partialN(3,5),
              f_partialN(2,5),
              f_partialN(0,5),
              f_partialN(1,5),
              f_partialN(4,5))

partial2 <- c(f_partialP(1,1,0,4),
              f_partialP(1,1,1,4),
              f_partialP(1,1,1,4),
              f_partialP(1,1,2,4),

              f_partialP(0,1,1,4),
              f_partialP(0,1,2,4),
              f_partialP(0,1,4,4),
```

```
              f_partialP(1,1,4,4),
              f_partialP(0,1,0,4))

names = c(    "Correct Answer",
              "Response",
              "i ",
              "Dichotomous",
              "Partial [-1/n, +1/n]",
              "Partial[-1/q, +1/p]")

dt <- data.frame(correct, response, i, abs, partial1 , partial2)

kbl(dt, col.names = names, caption = title, digits=3) %>%
  kable_classic() %>%
    add_header_above(c("Response Scenario " = 3, "Scores" = 3)) %>%
    pack_rows("Perfect Response", 1, 1) %>%
    pack_rows("Correct + Extra Incorrect Selections", 2, 4) %>%
    pack_rows("Only Incorrect Selections", 5, 6) %>%
    pack_rows("Completely Inverse Response ", 7, 7) %>%
    pack_rows("Selected ALL or NONE", 8, 9) %>%
    footnote(general = paste("i = number of options in correct state; _ indicates option n
           general_title = "Note: ",footnote_as_chunk = T)
```

```
#cleanup
rm(dt, abs, correct,i,names,partial1,partial2,response,title)
```

- We see that in the Dichotomous scheme, only the precisely correct response (row 1) yields a score other than zero. This scheme does now allow us to differentiate between different response patters.
- The Partial $[-1/n, +1/n]$ scheme yields a range from $[-1, 1]$, differentiating between responses. However, a blank response (bottom row) receives the same score (0.6) as the selection of the correct option + 1 incorrect option (row 2), which is problematic with for the goals of this study, where we need to differentiate between states of confusion or uncertainty yielding blank responses and the intentional selection of incorrect items.
- The Partial $[-1/q, +1/p]$ scheme also yields a range of scores from $[-1, 1]$. A blank response (bottom row) yields the same score (0) as the selection of *all* answer options (row 7); both are patterns of behavior we would expect to see if a respondent is confused or uncertain that there is a correct answer to the question.

Next we consider an example from our study, with $n = 15$ options and $p = 1$ correct option to be selected.

Table 2.2: Comparison of Scoring Schemes for n = 5 options [A,B,C,D,E]

Response Scenario			Scores		
Correct Answer	Response	i	Dichotomous	Partial [-1/n, +1/n]	Partial[-1/q, +1/p]
Perfect Response					
A_____	A_____	5	1	1.0	1.00
Correct + Extra Incorrect Selections					
A_____	AB____	4	0	0.6	0.75
A_____	A____E	4	0	0.6	0.75
A_____	AB___E	3	0	0.2	0.50
Only Incorrect Selections					
A_____	_____E	3	0	0.2	-0.25
A_____	____DE	2	0	-0.2	-0.50
Completely Inverse Response					
A_____	_BCDE	0	0	-1.0	-1.00
Selected ALL or NONE					
A_____	ABCDE	1	0	-0.6	0.00
A_____	_____	4	0	0.6	0.00

Note: i = number of options in correct state; _ indicates option not selected

```
title <- "Comparison of Scoring Schemes for SGC3 with n=15 and p=1 options [A,B...N,O]  "

correct <- c( "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____",
              "A____" )

response <- c("A__..__",
              "AB_..__",
              "A__..._O",
              "AB_..._O",
              "___..._O",
              "___...NO",
              "_BC...NO",
              "ABC...NO",
              "___..__" )
```

```
i <- c(         15,
                14,
                14,
                13,
                13,
                12,
                0,
                1,
                14)

abs <- c(f_dichom(15,15),
         f_dichom(14,15),
         f_dichom(14,15),
         f_dichom(13,15),
         f_dichom(13,15),
         f_dichom(12,15),
         f_dichom(0,15),
         f_dichom(1,15),
         f_dichom(14,15))

partial1 <- c(f_partialN(15,15),
              f_partialN(14,15),
              f_partialN(14,15),
              f_partialN(13,15),
              f_partialN(13,15),
              f_partialN(12,15),
              f_partialN(0,15),
              f_partialN(1,15),
              f_partialN(14,15))

partial2 <- c(f_partialP(1,1,0,14),
              f_partialP(1,1,1,14),
              f_partialP(1,1,1,14),
              f_partialP(1,1,2,14),
              f_partialP(0,1,1,14),
              f_partialP(0,1,2,14),
              f_partialP(0,1,14,14),
              f_partialP(1,1,14,14),
              f_partialP(0,1,0,14))

names = c(    "Correct Answer",
```

Table 2.3: Comparison of Scoring Schemes for SGC3 with n=15 and p=1 options [A,B...N,O]

Response Scenario			Scores		
Correct Answer	Response	i	Dichotomous	Partial [-1/n, +1/n]	Partial [-1/q, +1/p]
Perfect Response					
A_____	A___...___	15	1	1.000	1.000
Correct + Extra Incorrect Selections					
A_____	AB_...___	14	0	0.867	0.929
A_____	A___..._O	14	0	0.867	0.929
A_____	AB_..._O	13	0	0.733	0.857
Only Incorrect Selections					
A_____	_____..._O	13	0	0.733	-0.071
A_____	_____...NO	12	0	0.600	-0.143
Completely Inverse Response					
A_____	_BC...NO	0	0	-1.000	-1.000
Selected ALL or NONE					
A_____	ABC...NO	1	0	-0.867	0.000
A_____	_____..._____	14	0	0.867	0.000

Note: i = number of options in correct state; _ indicates option not selected

```
              "Response",
              "$i$ ",
              "Dichotomous",
              "Partial [-1/n, +1/n]",
              "Partial [-1/q, +1/p]")

dt <- data.frame(correct, response, i, abs, partial1 , partial2)

kbl(dt, col.names = names, caption = title, digits=3) %>%
  kable_classic() %>%
    add_header_above(c("Response Scenario " = 3, "Scores" = 3)) %>%
    pack_rows("Perfect Response", 1, 1) %>%
    pack_rows("Correct + Extra Incorrect Selections", 2, 4) %>%
    pack_rows("Only Incorrect Selections", 5, 6) %>%
    pack_rows("Completely Inverse Response ", 7, 7) %>%
    pack_rows("Selected ALL or NONE", 8, 9) %>%
    footnote(general = paste("i = number of options in correct state; _ indicates option n
           general_title = "Note: ",footnote_as_chunk = T)
```

```
#cleanup
rm(dt, abs, correct,i,names,partial1,partial2,response,title)
```

Here again we see that the Partial $[-1/q, +1/p]$ scheme allows us differentiate between patterns of responses, in a way that is more sensible for the goals of the SGC3 graph comprehension task.

> The Partial $[-1/q, +1/p]$ scheme is more appropriate for scoring the graph comprehension task than the Partial $[-1/n, +1/n]$ scheme because it allows us to differentially penalize incorrectly *selected* and incorrectly *not selected* answer options.

2.1.4 TODO A Discriminant Score

Though it appears the Partial $[-1/q, +1/p]$ scheme is a more appropriate scoring method for differentiating between meaningful patterns of responses, it does have one significant limitation: it treats all incorrectly selected answer options in the same way.

In the case of the SGC study paradigm, the choice of which options the subject selects reveals important information about their understanding of the graph stimulus. Specifically, there are certain patterns of options that correspond to a *triangular* versus *orthogonal* interpretations. In the previous example, rows 2 and 3 both indicate a subject has selected the correct option (A) plus one additional option (B or O). Both responses receive the same score under $[-1/q, +1/q]$. These two responses may be very meaningfully different, however if one of the alternate options chosen indicates a different interpretation of the stimulus. The inclusion of data-point B might be reasonable under some degree of visual tracing error, while the inclusion of data-point O might indicate a cartesian misinterpretation of the coordinate system. Essentially, some answer options are *more incorrect* than others. We want to be able to differentiate between these responses.

To accomplish this task, we need to sets of interpretation-consistent options for each item. Based on our prior work, we define **four interpretations** for which to define options. We can think of these as four different answer keys, each defining the set of 'correct' options (those that should be selected) under each interpretation of the graph.

1. Triangular : the (true, actually correct) triangular interpretation of the coordinate system; indicated by data points intersecting the appropriate diagonal projection from the x-axis.
2. Orthogonal: the (incorrect but most common) cartesian interpretation of the coordinate system; indicated by data points intersecting an (invisible) orthogonal projection from the x-axis.

3. Lines-Connect: an (incorrect) interpretation that is partially-triangular, insofar as it is indicated by data points intersecting any diagonal projection from the x-axis (not necessarily the *correct* projection).
4. Satisficing: an (incorrect) interpretation that indicated by selecting the data points nearest to the (invisible) orthogonal projection from the x-axis; indicates an orthogonal interpretation in absence of an available orthogonal intersect.

TODO FIX THIS SECTION ONWARDS … RESPONSE SETS NEED NOT BE MUTUTALLY EXCLUSIVE

Thus for *each* item, in the graph comprehension task, we will define five sets, whose members constitute a partition of the set of answer options.

$$Q = \text{the (universal) set of all answer options} \tag{2.16}$$
$$\tag{2.17}$$
$$T = \{o : o\, is\, triangular\}, |T| \neq 0 \tag{2.18}$$
$$R = \{o : o\, is\, orthogonal\} \tag{2.19}$$
$$L = \{o : o\, is\, line - connecting\} \tag{2.20}$$
$$S = \{o : o\, is\, satisficing\} \tag{2.21}$$
$$\tag{2.22}$$
$$\emptyset = T \cap R \cap L \cap S \,(\text{the interpretation sets are disjoint}) \tag{2.23}$$
$$D = Q - \{T \cup R \cup L \cup S\}, (\text{distractors; the remaining options not consistent with any interpretation}) \tag{2.24}$$
$$\tag{2.25}$$

For example, for the following sample stimuli, TODO IMAGE

$$Q = \{a, b, c, d, e, f, g\} \tag{2.26}$$
$$= \{\{a\}\}, \{b\}, \{c, d\}, \{e\}, \{f, g\}\} \tag{2.27}$$
$$\tag{2.28}$$
$$T = \{a\} \tag{2.29}$$
$$R = \{b\} \tag{2.30}$$
$$L = \{c, d\} \tag{2.31}$$
$$S = \{e\} \tag{2.32}$$
$$D = \{f, g\} \tag{2.33}$$
$$\tag{2.34}$$

To capture this important source of variation, we can apply the idea behind the partial scoring $[-1/q, +1/p]$ scheme in combination with the response option subgroups given by the graph interpretations to produce a single score (scaled from -1 to $+1$) that captures the *nature* of the respondent's partial knowledge.

A **Discriminant Score** will offer:

- +1 for a complete triangular response
- -1 for a complete orthogonal response
- +/- fraction of complete triangular or orthogonal response
- 0 for non-strategy responses
- 0 for blank (empty) responses

$$\text{Score}_{discriminant} = 0 + 1 * (t_s/t) - 1 * (r_s/r) + 0 * (d_s/d) \tag{2.35}$$
$$= (t_s/t) - (r_s/r) \tag{2.36}$$

Where:

- $t = |T|$ number of triangular-correct options (consistent with triangular interpretation)
- $r = |R|$ = number of orthogonal-correct options (consistent with orthogonal interpretation)
- d = number of non-strategy options (not consistent with either interpretation)
- n = total number of response options (equals the number of checkboxes in the question; equals the number of data points in the stimulus graph)
- $n = t + r + d$ (15 for scaffold phase, 18 for test phase)
- $t_s \geq 0$ number of triangular-correct options selected
- $r_s \geq 0$ = number of orthogonal-correct options selected
- $d_s \geq 0$ = number of non-strategy options options selected
- $s \geq 0$ = number of options selected
- $s = t_s + r_s + d_s$

TODO:: add 0.5*(tversky) - 0.5*(satisficing)

```
f_discrim <- function(t_s,t,r_s,r){
  return((t_s / t) - (r_s / r))
}
```

2.2

2.3 SCORE SGC DATA

In SGC_3A we are fundamentally interested in understanding how a participant interprets the presented graph (stimulus). The graph comprehensiontask asks them to select the data points in the graph that meet the criteria posed in the question. To assess a participant's performance, for each question (q=15) we will calculate the following scores:

An overall, strict score:
1. **Absolute Score** : using dichotomous scoring referencing true (Triangular) answer. (see 1.2)

Sub-scores, for each alternative graph interpretation
2. **Triangular Score** : using partial scoring [-1/q, +1/p] referencing true (Triangular) answer key.

3. **Orthogonal Score** : using partial scoring [-1/q, +1/p] referencing (incorrect Orthogonal) answer key.

4. **Tversky Score** : using partial scoring [-1/q, +1/p] referencing (incorrect connecting-lines strategy) answer key.

5. **Satisficing Score** : using partial scoring [-1/q, +1/p] referencing (incorrect satisficing stategy) answer key.

2.3.1 Prepare Answer Keys

We start by importing the answer keys for each stimulus set. For each stimulus set (defined by condition+task phase) an answer key file exist that defines the set of response options that indicate each graph interpretation.

```
key_111_raw <- read_csv('keys/SGC3A_scaffold_111_key.csv') %>% mutate(condition = 111, pha
key_121_raw <- read_csv('keys/SGC3A_scaffold_121_key.csv')%>% mutate(condition = 121, phas
key_test_raw <- read_csv('keys/SGC3A_test_key.csv')%>% mutate(condition = "", phase = "tes

#TODO decompose tversky responses for key_121_raw and key_test_raw

keys_raw <- rbind(key_111_raw) #, key_121_raw, key_test_raw)
rm(key_111_raw, key_121_raw, key_test_raw)
```

In order to calculate scores using the $[-1/q, +1/p]$ algorithm, we need to define the subset of all response options (set N) that should be selected (set P) and should not be selected (set

Q). In order to calculate subscores for each graph interpretation (triangular, orthogonal) we must define these sets independently for each interpretation. The answer key files the 'correct' answer (i.e. set P) for each interpretation, for each question. We must derive the contents of set Q based on set

- SET N = all response options (superset) N
- SET P = all options that should be selected (rewarded as +1/p) $P \subset N$
- SET A = items that should not be selected, but if they are, don't penalize (ignored). These include any options referenced in the question (i.e. select shifts that start at the same time as X; don't penalize if they also select 'X'), as well as options within 0.5hr offset from the data point to accommodate reasonable visual errors.$A \subset N$
- SET Q = all options that should NOT be selected (penalized as -1/q). These are options that weren't referenced in the question and aren't within 0.5hr of the correct option. $Q = N - (P \cup A)$

2.3.1.1 Triangular Key

First we construct the key set based on the 'Triangular' interpretation (ie. the actually correct answers). TODO EXAMPLE IMAGE

```
verify = c() #sanity check
##------------------------------------------------------------------------
##CONSTRUCT TRIANGULAR KEY SET
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_tri <- keys_raw %>%
  select(Q, condition, OPTIONS, TRIANGULAR, TRIve, ALSO) %>%
  mutate(

    #replace NAs
    TRIve = str_replace_na(TRIve,""),
    ALSO = str_replace_na(ALSO,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TRIANGULAR,
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TRIve,ALSO, sep=""),
    n_a = nchar(set_a),
```

```
    #N store all answr options (superset)
    set_n = OPTIONS,
    n_n = nchar(set_n)

) %>% select(Q, condition, set_n, set_p, set_a, n_n, n_p, n_a)

#2. DEFINE SETS N, P, A
for (x in 1:nrow(keys_tri)) {

  #UNWIND STRINGS FOR SETDIFF
  #n all answer options
  N = keys_tri[x,'set_n'] %>% pull(set_n) %>% strsplit("") %>% unlist()
  #p correct-select answer options
  P = keys_tri[x,'set_p'] %>% pull(set_p) %>% strsplit("") %>% unlist()
  #a ignore-select answer options (should not be selected, but if they are, don't penalize
  A = keys_tri[x,'set_a'] %>% pull(set_a) %>% strsplit("") %>% unlist()

  #Q = N - (P+A)
  #answers that are penalized (at -1/q) if selected
  s = union(P,A) #rewarded plus ignored
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected

  #save set to dataframe
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_tri[x,'set_q'] = Q
  keys_tri[x,'n_q'] = n_q

  #verify each element in N is included in one and only one of P,A,Q
  tempunion = union(s,Q) %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  N = N %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  verify[x] = setequal(tempunion,N)
}

#3. reorder cols for ease of use
keys_tri <- keys_tri %>% select(Q, condition, set_n, set_p, set_a, set_q, n_n, n_p, n_a, n

#cleanup
rm(N,A,P,Q,n_q,s,x,tempunion)
```

This leaves us a dataframe `keys_tri` that define the sets of response options consistent with the triangular graph interpretation.

To verify we have generated the correct sets, we verify that for each row (question), each response in N is included in either set P, A or Q (once and only once).

TRUE, TRUE, TRUE, TRUE, TRUE TODO TROUBLESHOOT FALSE; THINK THIS IS BECAUSE KEY FILE IS INCOMPLETE FOR ALL BUT RAW 111

2.3.1.2 Orthogonal Key

TODO EXAMPLE IMAGE Next we construct the key set based on the 'Orthogonal' interpretation.

```
verify = c() #sanity check

##------------------------------------------------------------------------
##CONSTRUCT ORTHOGONAL KEY SET
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_orth <- keys_raw %>%
  select(Q, condition, OPTIONS, ORTHOGONAL, ORTHve, ALSO) %>%
  mutate(

    #replace NAs
    ORTHve = str_replace_na(ORTHve,""),
    ALSO = str_replace_na(ALSO,""),

    #P options that SHOULD be selected (rewarded)
    set_p = ORTHOGONAL,
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(ORTHve,ALSO, sep=""),
    n_a = nchar(set_a),

    #N store all answr options (superset)
    set_n = OPTIONS,
    n_n = nchar(set_n)

) %>% select(Q, condition, set_n, set_p, set_a, n_n, n_p, n_a)
```

```
#2. DO THE STUFF THAT'S EASIER IN A LOOP
for (x in 1:nrow(keys_orth)) {

  #UNWIND STRINGS FOR SETDIFF
  #n all answer options
  N = keys_orth[x,'set_n'] %>% pull(set_n) %>% strsplit("") %>% unlist()
  #p correct-select answer options
  P = keys_orth[x,'set_p'] %>% pull(set_p) %>% strsplit("") %>% unlist()
  #a ignore-select answer options (should not be selected, but if they are, don't penalize
  A = keys_orth[x,'set_a'] %>% pull(set_a) %>% strsplit("") %>% unlist()

  #Q = N - (P+A)
  #answers that are penalized (at -1/q) if selected
  s = union(P,A) #rewarded plus ignored
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected

  #save set to dataframe
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_orth[x,'set_q'] = Q
  keys_orth[x,'n_q'] = n_q

  #verify each element in N is included in one and only one of P,A,Q
  tempunion = union(s,Q) %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  N = N %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  verify[x] = setequal(tempunion,N)
}

#3. reorder cols for ease of use
keys_orth <- keys_orth %>% select(Q, condition, set_n, set_p, set_a, set_q, n_n, n_p, n_a,

#cleanup
rm(A, N, n_q, P, Q, s, tempunion, x)
```

This leaves us a dataframe **keys_orth** that define the sets of response options consistent with the orthogonal graph interpretation.

To verify we have generated the correct sets, we verify that for each row (question), each response in N is included in either set P, A or Q (once and only once).

TRUE, TRUE, TRUE, TRUE, TRUE TODO TROUBLESHOOT FALSE; THINK THIS IS BECAUSE KEY FILE IS INCOMPLETE FOR ALL BUT RAW 111

2.3.1.3 Tversky Keys

TODO EXAMPLE IMAGE Next we construct the key set based on the 'Tversky' interpretation. The term Tversky is shorthand for a partial interpretation of the coordinate system where subjects select a set of responses that lay along a connecting line from the referenced data point or referenced time for that item. The term is named for Barbara Tversky based on her work on graphical primitives (e.g. "lines connect").

```
verify_max = c() #sanity check
##------------------------------------------------------------------------
##CONSTRUCT TVERSKY KEY SET for TVERSKY-MAX
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_tversky_max <- keys_raw %>%
  select(Q, condition, OPTIONS, ALSO, TV_max, TV_max_ve) %>%
  mutate(

    #replace NAs
    ALSO = str_replace_na(ALSO,""),
    TV_max = str_replace_na(TV_max,""),
    TV_max_ve = str_replace_na(TV_max_ve,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TV_max,
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TV_max_ve,ALSO, sep=""),
    n_a = nchar(set_a),

    #N store all answr options (superset)
    set_n = OPTIONS,
    n_n = nchar(set_n)

) %>% select(Q, condition, set_n, set_p, set_a, n_n, n_p, n_a)

#2. DO THE STUFF THAT'S EASIER IN A LOOP
for (x in 1:nrow(keys_tversky_max)) {

  #UNWIND STRINGS FOR SETDIFF
  #n all answer options
  N = keys_tversky_max[x,'set_n'] %>% pull(set_n) %>% strsplit("") %>% unlist()
```

```
  #p correct-select answer options
  P = keys_tversky_max[x,'set_p'] %>% pull(set_p) %>% strsplit("") %>% unlist()
  #a ignore-select answer options (should not be selected, but if they are, don't penalize
  A = keys_tversky_max[x,'set_a'] %>% pull(set_a) %>% strsplit("") %>% unlist()

  #Q = N - (P+A)
  #answers that are penalized (at -1/q) if selected
  s = union(P,A) #rewarded plus ignored
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected

  #save set to dataframe
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_tversky_max[x,'set_q'] = Q
  keys_tversky_max[x,'n_q'] = n_q

  #verify each element in N is included in one and only one of P,A,Q
  tempunion = union(s,Q) %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  N = N %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  verify_max[x] = setequal(tempunion,N)
}

#3. reorder cols for ease of use
keys_tversky_max <- keys_tversky_max %>% select(Q, condition, set_n, set_p, set_a, set_q,

verify_tversky_start = c() #sanity check
##------------------------------------------------------------------------
##CONSTRUCT TVERSKY KEY SET for TVERSKY-START
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_tversky_start <- keys_raw %>%
  select(Q, condition, OPTIONS, ALSO, TV_start, TV_start_ve) %>%
  mutate(

    #replace NAs
    ALSO = str_replace_na(ALSO,""),
    TV_start = str_replace_na(TV_start,""),
    TV_start_ve = str_replace_na(TV_start_ve,""),

    #P options that SHOULD be selected (rewarded)
```

```
    set_p = TV_start,
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TV_start_ve,ALSO, sep=""),
    n_a = nchar(set_a),

    #N store all answr options (superset)
    set_n = OPTIONS,
    n_n = nchar(set_n)

) %>% select(Q, condition, set_n, set_p, set_a, n_n, n_p, n_a)

#2. DO THE STUFF THAT'S EASIER IN A LOOP
for (x in 1:nrow(keys_tversky_start)) {

  #UNWIND STRINGS FOR SETDIFF
  #n all answer options
  N = keys_tversky_start[x,'set_n'] %>% pull(set_n) %>% strsplit("") %>% unlist()
  #p correct-select answer options
  P = keys_tversky_start[x,'set_p'] %>% pull(set_p) %>% strsplit("") %>% unlist()
  #a ignore-select answer options (should not be selected, but if they are, don't penalize
  A = keys_tversky_start[x,'set_a'] %>% pull(set_a) %>% strsplit("") %>% unlist()

  #Q = N - (P+A)
  #answers that are penalized (at -1/q) if selected
  s = union(P,A) #rewarded plus ignored
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected

  #save set to dataframe
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_tversky_start[x,'set_q'] = Q
  keys_tversky_start[x,'n_q'] = n_q

  #verify each element in N is included in one and only one of P,A,Q
  tempunion = union(s,Q) %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  N = N %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  verify_tversky_start[x] = setequal(tempunion,N)
}
```

```
#3. reorder cols for ease of use
keys_tversky_start <- keys_tversky_start %>% select(Q, condition, set_n, set_p, set_a, set


verify_tversky_end = c() #sanity check
##-----------------------------------------------------------------------
##CONSTRUCT TVERSKY KEY SET for TVERSKY-END
##-----------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_tversky_end <- keys_raw %>%
  select(Q, condition, OPTIONS, ALSO, TV_end, TV_end_ve) %>%
  mutate(

    #replace NAs
    ALSO = str_replace_na(ALSO,""),
    TV_end = str_replace_na(TV_end,""),
    TV_end_ve = str_replace_na(TV_end_ve,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TV_end,
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TV_end_ve,ALSO, sep=""),
    n_a = nchar(set_a),

    #N store all answr options (superset)
    set_n = OPTIONS,
    n_n = nchar(set_n)

) %>% select(Q, condition, set_n, set_p, set_a, n_n, n_p, n_a)

#2. DO THE STUFF THAT'S EASIER IN A LOOP
for (x in 1:nrow(keys_tversky_end)) {

  #UNWIND STRINGS FOR SETDIFF
  #n all answer options
  N = keys_tversky_end[x,'set_n'] %>% pull(set_n) %>% strsplit("") %>% unlist()
  #p correct-select answer options
  P = keys_tversky_end[x,'set_p'] %>% pull(set_p) %>% strsplit("") %>% unlist()
  #a ignore-select answer options (should not be selected, but if they are, don't penalize
```

```
  A = keys_tversky_end[x,'set_a'] %>% pull(set_a) %>% strsplit("") %>% unlist()

  #Q = N - (P+A)
  #answers that are penalized (at -1/q) if selected
  s = union(P,A) #rewarded plus ignored
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected

  #save set to dataframe
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_tversky_end[x,'set_q'] = Q
  keys_tversky_end[x,'n_q'] = n_q

  #verify each element in N is included in one and only one of P,A,Q
  tempunion = union(s,Q) %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  N = N %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  verify_tversky_end[x] = setequal(tempunion,N)
}

#3. reorder cols for ease of use
keys_tversky_end <- keys_tversky_end %>% select(Q, condition, set_n, set_p, set_a, set_q,

verify_tversky_duration = c()
##------------------------------------------------------------------------
##CONSTRUCT TVERSKY KEY SET for TVERSKY-DURATION
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_tversky_duration <- keys_raw %>%
  select(Q, condition, OPTIONS, ALSO, TV_dur, TV_dur_ve) %>%
  mutate(

    #replace NAs
    ALSO = str_replace_na(ALSO,""),
    TV_dur = str_replace_na(TV_dur,""),
    TV_dur_ve = str_replace_na(TV_dur_ve,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TV_dur,
    n_p = nchar(set_p), #number of true-select options
```

```
    #A options that are ignored if selected
    set_a = str_c(TV_dur_ve,ALSO, sep=""),
    n_a = nchar(set_a),

    #N store all answr options (superset)
    set_n = OPTIONS,
    n_n = nchar(set_n)

) %>% select(Q, condition, set_n, set_p, set_a, n_n, n_p, n_a)

#2. DO THE STUFF THAT'S EASIER IN A LOOP
for (x in 1:nrow(keys_tversky_duration)) {

  #UNWIND STRINGS FOR SETDIFF
  #n all answer options
  N = keys_tversky_duration[x,'set_n'] %>% pull(set_n) %>% strsplit("") %>% unlist()
  #p correct-select answer options
  P = keys_tversky_duration[x,'set_p'] %>% pull(set_p) %>% strsplit("") %>% unlist()
  #a ignore-select answer options (should not be selected, but if they are, don't penalize
  A = keys_tversky_duration[x,'set_a'] %>% pull(set_a) %>% strsplit("") %>% unlist()

  #Q = N - (P+A)
  #answers that are penalized (at -1/q) if selected
  s = union(P,A) #rewarded plus ignored
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected

  #save set to dataframe
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_tversky_duration[x,'set_q'] = Q
  keys_tversky_duration[x,'n_q'] = n_q

  #verify each element in N is included in one and only one of P,A,Q
  tempunion = union(s,Q) %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  N = N %>% str_c(collapse="") %>% strsplit("") %>% unlist()
  verify_tversky_duration[x] = setequal(tempunion,N)
}

#3. reorder cols for ease of use
keys_tversky_duration <- keys_tversky_duration %>% select(Q, condition, set_n, set_p, set_
```

```
#cleanup
rm(A, N, n_q, P, Q, s, tempunion, x)
```

This leaves us four dataframes, each corresponding to a different variant of a 'lines connecting to reference point' strategy.
- `keys_tversky_max` : the superset of lines connecting options - `keys_tversky_start` : lines connecting to the rightward diagonal (start time) of the reference point - `keys_tversky_end`: lines connecting to the leftward diagonal (end time) of the reference point - `keys_tversky_duration`: lines connecting to the horizontal y-intercept (duration) of the reference point

To verify we have generated the correct sets, we verify that for each row (question), each response in N is included in either set P, A or Q (once and only once).

TRUE, TRUE, TRUE, TRUE, TRUE TODO TROUBLESHOOT FALSE; THINK THIS IS BECAUSE KEY FILE IS INCOMPLETE FOR ALL BUT RAW 111 TRUE, TRUE, TRUE, TRUE, TRUE TRUE, TRUE, TRUE, TRUE, TRUE TRUE, TRUE, TRUE, TRUE, TRUE

```
#cleanup
rm(verify, verify_max, verify_tversky_duration, verify_tversky_end, verify_tversky_start)
```

2.3.2 Score Items

Next, we import the item-level response data. For each row in the item level dataset (indicating the response to a single question-item for a single subject), we compare the raw response `df_items$response` with the answer keys in each interpretation (keys_orth, keys_tri), then using those sets, determine the number of correctly selected items(p) and incorrectly selected items (q), which in turn are used to calculate partial[-1/q, +1/p] scores for each interpretation. The resulting scores are then stored on each item, and can be used to determine which graph interpretation the subject held.
`score_TRI` (how triangular was the subject's response?) `score_ORTH`(how orthogonal was the subject's response)

```
backup <- read_rds('data/sgc3a_items.rds')
df_items <- read_rds('data/sgc3a_items.rds')
df_items <- df_items %>% filter(q < 6) %>% filter(condition == 111)
```

note: this cell takes several minutes to run and is not optimized

```
#CALCULATE THE TRIANGULAR, ORTHOGONAL OR TVERSKIAN SUBSCORES FROM KEYFRAME
calc_sub_score <- function(keyframe, question, condition, response){

  #TODO verify type is in df_keymap$score
  #type must be one of "tri", "orth", "tv_max", "tv_start", "tv_end", "tv_duration", "both
  #keyframe must be one of keys_raw, keys_tri, "keys_orth", "keys_tversky_max", "keys_tver

  #STEP 1 GET KEY
  if (question < 6) #for q1 - q5 find key for question by condition
  {
    # print(keyframe)
    #GET KEY FOR THIS SCORE TYPE, QUESTION AND CONDITION
    p =  keyframe %>% filter(Q == question) %>% filter(condition == condition) %>% select(
    q =  keyframe %>% filter(Q == question) %>% filter(condition == condition) %>% select(
    pn = keyframe %>% filter(Q == question) %>% filter(condition == condition) %>% select(
    qn = keyframe %>% filter(Q == question) %>% filter(condition == condition) %>% select(

  } else {
    #GET KEY FOR THIS SCORE TYPE, QUESTION
    p =  keyframe %>% filter(Q == question) %>% select(set_p) %>% pull(set_p) %>% str_spli
    q =  keyframe %>% filter(Q == question) %>% select(set_q) %>% pull(set_q) %>% str_spli
    pn = keyframe %>% filter(Q == question) %>% select(n_p)
    qn = keyframe %>% filter(Q == question) %>% select(n_q)
  }

  #STEP 2 CALC INTERSECTIONS BETWEEN RESPONSE AND KEY
  ps = length(intersect(r,p))
  qs = length(intersect(r,q))
  # df_items[x,'tri_ps'] = tri_ps
  # df_items[x,'tri_qs'] = tri_qs

  #STEP 3 CALC f_partialP schema SCORE FOR THIS INTERSECTION
  x = f_partialP(ps,pn,qs,qn)

  #cleanup
  rm(p,q,pn,qn,ps,qs)

  return(x) #true correct, trues, false correct, falses

}

#CALCULATE THE REFERENCE SCORES
```

```
calc_ref_score <- function(question, condition, response){

  #STEP 1 GET KEY for this question and condition
  if (question < 6) #for q1 - q5 find key for question by condition
  {
     #1. GET reference point from ALSO column in raw keys
     ref_p = keys_raw %>% filter(Q == question) %>% filter(condition == condition) %>% sel
     #2. get all possible options [doesn't matter which key, they're all the same]
     ref_n =keys_raw %>% filter(Q == question) %>% filter(condition == condition) %>% sele
     #3. calc answers that aren't the reference point
     ref_q = setdiff(ref_n,ref_p)
     #4. cal number of ps and qs
     ref_pn = length(ref_p)
     ref_qn = length(ref_q)
  } else {
    #get REFERENCE POINT keys for this question
     #1. GET reference point from ALSO column in raw keys
     ref_p = keys_raw %>% filter(Q == question) %>% select(ALSO) %>% pull(ALSO) %>% str_sp
     #2. get all possible options [doesn't matter which key, they're all the same]
     ref_n =keys_raw %>% filter(Q == question) %>% select(OPTIONS) %>% pull(OPTIONS) %>% s
     #3. calc answers that aren't the reference point
     ref_q = setdiff(ref_n,ref_p)
     #4. cal number of ps and qs
     ref_pn = length(ref_p)
     ref_qn = length(ref_q)
  }

  #STEP 2 CALC INTERSECTIONS BETWEEN RESPONSE AND KEY
  ref_ps = length(intersect(response,ref_p))
  ref_qs = length(intersect(response,ref_q))
  # df_items[x,'both_ps'] = ref_ps
  # df_items[x,'both_qs'] = ref_qs

  #STEP 3 CALC f_partialP schema SCORE FOR THIS INTERSECTION
  x = f_partialP(ref_ps,ref_pn,ref_qs,ref_qn)

  #cleanup
  rm(ref_p,ref_q,ref_pn,ref_qn,ref_ps,ref_qs)

  return(x) #true correct, trues, false correct, falses
```

```
}


#CALCULATE SCORE BASED ON UNION OF ORTH & TRI (SUBJECT SELECTS BOTH ANSWERS )
calc_both_score <- function(question, condition, response){

  #STEP 1 GET KEY for this question and condition
  if (question < 6) #for q1 - q5 find key for question by condition
  {
     #grab the tri and orth keys for this question as well as N option set
     tri_p =  keys_tri %>%  filter(Q == question) %>% filter(condition == condition) %>% s
     orth_p = keys_orth %>% filter(Q == question) %>% filter(condition == condition) %>% s
     set_n =  keys_tri %>%  filter(Q == question) %>% filter(condition == condition) %>% s
     #1. calc answer that is both tri and orth and only these --> union of tri_p and orth_
     both_p = union(tri_p, orth_p) #the selection of tri and p
     #2. calc answers that should't be selected as diffrence between N [same for all keys]
     both_q = setdiff(set_n,both_p)
     both_pn = length(both_p)
     both_qn = length(both_q)
  } else{

     #grab the tri and orth keys for this question as well as N option set
     tri_p =  keys_tri %>%  filter(Q == question) %>% select(set_p) %>% pull(set_p) %>% st
     orth_p = keys_orth %>% filter(Q == question) %>% select(set_p) %>% pull(set_p) %>% st
     set_n =  keys_tri %>%  filter(Q == question) %>% select(set_n) %>% pull(set_n) %>% st
     #1. calc answer that is both tri and orth and only these --> union of tri_p and orth_
     both_p = union(tri_p, orth_p) #the selection of tri and p
     #2. calc answers that shouldn't be selected as difference between N [same for all key
     both_q = setdiff(set_n,both_p)
     both_pn = length(both_p)
     both_qn = length(both_q)
  }

  #STEP 2 CALC INTERSECTIONS BETWEEN RESPONSE AND KEY
  both_ps = length(intersect(response,both_p))
  both_qs = length(intersect(response,both_q))
  # df_items[x,'both_ps'] = both_ps
  # df_items[x,'both_qs'] = both_qs

  #STEP 3 CALC f_partialP schema SCORE FOR THIS INTERSECTION
  x = f_partialP(both_ps,both_pn,both_qs,both_qn)
```

```
  #cleanup
  rm(both_p,both_q,both_pn,both_qn,both_ps,both_qs)

  return(x) #true correct, trues, false correct, falses

}
```

```
#CALCULATE SUBSCORES (in loop)
for (x in 1:nrow(df_items)) {

  #get properties of the RESPONSE ITEM
  qu = df_items[x,'q']
  cond = df_items[x,'condition']
  r = df_items[x,'response'] %>% str_split("") %>% unlist()

  #calculate the main subscores
  df_items[x,'score_TRI'] = calc_sub_score(keys_tri, qu, cond, r)
  df_items[x,'score_ORTH'] = calc_sub_score(keys_orth, qu, cond, r)
  df_items[x,'score_TV_max'] = calc_sub_score(keys_tversky_max, qu, cond, r)
  df_items[x,'score_TV_start'] = calc_sub_score(keys_tversky_start, qu, cond, r)
  df_items[x,'score_TV_end'] = calc_sub_score(keys_tversky_end, qu, cond, r)
  df_items[x,'score_TV_duration'] = calc_sub_score(keys_tversky_duration, qu, cond, r)

  # #calculate special subscores
  df_items[x,'score_REF'] = calc_ref_score(qu, cond, r)
  df_items[x,'score_BOTH'] = calc_both_score(qu, cond, r)
}

#CALCULATE ABSOLUTE SCORES
#calculate absolute scores dichotomous
df_items$score_ABS = as.integer(df_items$correct)

#niceABS indicates if the response is correct without penalizing the allowable triangular
df_items$score_niceABS  <- as.integer((df_items$score_TRI == 1))

#niceABS indicates if the response is correct without penalizing the allowable triangular
# df_items[x,'score_niceABS'] = (df_items[x,'score_TRI'] == 1)
# df_items[x,'score_niceABS'] = as.integer(df_items[x,'score_niceABS'])

#cleanup
```

```
rm(qu,cond,r, x)
```

2.3.3 Derive Interpretation

Finally, we compare the interpretation sub-scores and decide which is highest. This indicates the interpretation that the individual's response most closely approximates. TODO HANDLE EQUAL COLUMNS TODO HANDLE BLANKS TODO HANDLE CONFUSION

1. is it blank –> blank
2. is it triangular?
3. is it orthogonal?
4. is it tversky?
5. is it satisfice?

```
df_items$best <- "?" #set to null
threshold_range = 0.5 #set required variance in subscores to be discriminant
threshold_frenzy = 7

# df_items <- backup
for (x in 1:nrow(df_items)) {

  #CALCULATE MAX TVERSKY SUBSCORE
  #first reshape subscores
  t = df_items[x,] %>% select(score_TV_max, score_TV_start, score_TV_end, score_TV_duratio
  t.long = gather(t,score, value, 1:4)

  #replace empty scores with NA so we can ignore them
  #sometimes TVERSKY subscores are blank if they don't apply for that question
  t.long[t.long == ""] = NA

  df_items[x,'score_TVERSKY'] = as.numeric(max(t.long$value,na.rm = TRUE))
  df_items[x,'tv_type'] = t.long[which.max(t.long$value),'score']


  #NOW CALCULATE VARIANCE IN SUBSCORES
  s = df_items[x,] %>% select(score_TRI, score_ORTH, score_TVERSKY)
  s.long = gather(s,score, value, 1:3)

  #calculate the range between highest and lowest scores
  r = as.numeric(range(s.long$value,na.rm = TRUE))
```

```
  r = diff(r)
  df_items[x,'range'] = r
  # print(s.long)

  #DISCRIMINANT BETWEEN SUBSCORES TO PREDICT BEST FIT INTERPRETATION

  #is the response BLANK?
  if( df_items[x,'num_o'] == 0) {df_items[x,'best'] = "blank"}

  #is the response a 'frenzy'? ie. they selected more than half of the datapoints?
  else if( df_items[x,'num_o'] > threshold_frenzy) {df_items[x,'best'] = "frenzy"}

  #otherwise does the response perfectly match both TRI AND ORTH?
  else if( df_items[x,'score_BOTH'] == 1) {df_items[x,'best'] = "both tri + orth"}

  #otherwise does the response perfectly match the reference point
  else if( df_items[x,'score_REF'] == 1) {df_items[x,'best'] = "reference"}

  #otherwise try to derive from subscores
  else {

   #is there enough variance between scores to be able to differentiate?

    if (r < threshold_range) {
      #then we can't predict the interpretation
    }
    else {
      #if there IS enough variance to be predictive, take the highest subscore
      p = s.long[which.max(s.long$value),'score']
      if (p == "score_TRI") {df_items[x,'best'] = "Triangular"}
      else if (p == "score_ORTH") {df_items[x,'best'] = "Orthogonal"}
      else if (p == "score_TVERSKY") {df_items[x,'best'] = "Tversky"}

    }
  }


#REFERENCE CODE FOR FINDING MAX COLUMNS [to avoid reshaping dataframe to long]
# max = max.col(df_items[x,] %>% select(score_TRI,score_ORTH, score_TV_max, score_TV_start
# df_items[x,'best'] = names[max]
```

```
}

#cleanup
rm(t, t.long, s, s.long, x, r,p)
rm(threshold_frenzy, threshold_range)

#set order of levels
df_items$interpretation <- factor(df_items$best,
                                  levels = c("Orthogonal", "Triangular", "Tversky",
                                             "reference", "both tri + orth", "blank","fren

#recode as numeric
df_items$score_TV_duration <- df_items$score_TV_duration %>% as.numeric()
```

TODO is there still a need for a trapdoor or discriminating score?

```
f_partialC <- function(ps,p,qs,q) {
                       # gs,g) {

  #penalty_q = penalty for selecting an incorrect option
  #penalty_g = penalty for selecting an irrelevant option

  #ps = number of correct-selected options
  #p = number of true options

  #qs = number of incorrect-selected options
  #q = number of false options


  #gs = number of irrelevantly selected options
  #g = number of irrelevant options


  #n = number of options + p + q
  # return( (ps/p) - (qs/q) )
}
```

```
# test <- df_items %>% filter(q<6) %>% filter(condition == 111) %>% select(q, response, sc

# gf_histogram(~score_TRI, data = test)
# gf_histogram(~score_ORTH, data = test)
```

```
# gf_histogram(~score_ABS, data = test)
# gf_histogram(~score_niceABS, data = test)
```

2.4 EXPLORE RESPONSES

In this section we explore responses given by participants to each particular item in the graph comprehension task, and indicate how each response was scored, and what *interpretation of the graph* is indicated by different responses.

2.4.1 Control Condition

2.4.1.1 Question #1

We start by exploring the range of response options checked by participants on Question 1, for those assigned to the control (non-impasse) condition (`condition` = 111).

```
q <- keys_raw %>% filter(condition == 111) %>% filter(Q==1)
ignore <- q %>% select("ALSO")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL","TV_max","TV_start", "TV_end", "TV_dur"
ves <- q %>% select("TRIve", "ORTHve","TV_max_ve","TV_start_ve","TV_end_ve", "TV_dur_ve")%
options <- q %>% select("OPTIONS")
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Tversky [maximal]", "Tversky [start diagonal]",
            "Tversky [end diagonal]", "Tversky [duration line]")
d = tibble(interpretation = scores, answer = answers, allowed=ves)
d$answer <- replace_na(d$answer, "")
d$allowed <- replace_na(d$allowed, "")

title = paste("Answer Key | Q1 Control Condition : ", question)
cols = c("interpretation", "answer","not penalized")

d %>% kbl(caption = title, col.names = cols) %>% kable_classic() %>%
  footnote(general = paste("15 response options: ", options), general_title = "Note: ",foo
```

Notice that for this Question, the *Triangular* answer is the same as the *Tversky [start diagonal]* answer. In fact, for most questions, one of the Tversky sub-types will match the correct response.

Here we summarize the distinct response options given by participants on this item. Each letter in `response` indicates a checkbox selected by the participant (See Figure 2.3). `n` indicates

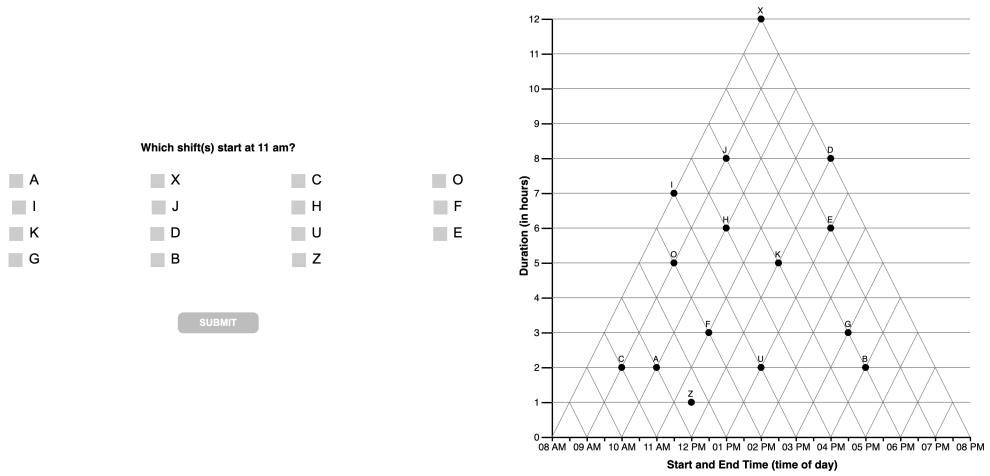


Figure 2.3: Question 1 — Control Condition

Table 2.4: Answer Key | Q1 Control Condition : Which shift(s) start at 11 am?

interpretation	answer	not penalized
Triangular	F	Z
Orthgonal	A	OI
Tversky [maximal]	CF	Z
Tversky [start diagonal]	F	Z
Tversky [end diagonal]	C	
Tversky [duration line]		

Note: 15 response options: AIKGXJDBCHUZOFE

the number of participants who gave this response, while `interpretation` indicates the *graph intepretation* most consistent with that response. At the right of this table are the Absolute, followed by Partial Credit subscores for each response.

```
title <- "Frequency of Selected Response Options for Question #1 (Control Condition)"
names = c("response","n","interpretation","strict","nice","tri","orth","tv_max","tv_start"

df_items %>% filter(q == 1 & condition == 111) %>% group_by(response) %>%
  dplyr::summarise( count = n(),
                    strict = unique(score_ABS),
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
                    orthogonal =  unique(score_ORTH),
                    tversky_max = unique(score_TV_max),
                    tversky_start = unique(score_TV_start),
                    tversky_end = unique(score_TV_end),
                    tversky_duration = unique(score_TV_duration),
                    interpretation = unique(interpretation)) %>%
  arrange(interpretation, desc(count)) %>%
  select(response, count, interpretation, strict, nice, triangular, orthogonal,
         tversky_max, tversky_start, tversky_end, tversky_duration) %>%
  kbl(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Scores" = 2, "Interpretation Scores"=6)) %>%
  pack_rows("Orthogonal", 1, 1) %>%
  pack_rows("Triangular", 2, 2) %>%
  pack_rows("Lines-Connect", 3, 3) %>%
  pack_rows("other", 4, 7)
```

\begin{table}

\caption{Frequency of Selected Response Options for Question #1 (Control Condition)}

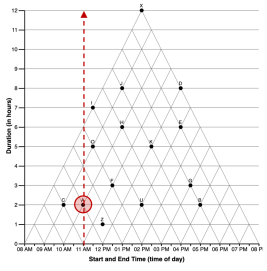
			Strict Scores		Interpretation Scores					
response	n	interpretation	strict	nice	tri	orth	tv_max	tv_start	tv_end	tv_dur
Orthogonal										
A	129	Orthogonal	0	0	-0.077	1.000	-0.083	-0.077	-0.071	NA
Triangular										
F	22	Triangular	1	1	1.000	-0.083	0.500	1.000	-0.071	NA
Lines-Connect										
CF	3	Tversky	0	0	0.923	-0.167	1.000	0.923	0.929	NA
other										
AF	1	both tri + orth	0	0	0.923	0.917	0.417	0.923	-0.143	NA
IJD	1	?	0	0	-0.231	-0.167	-0.250	-0.231	-0.214	NA
X	1	?	0	0	-0.077	-0.083	-0.083	-0.077	-0.071	NA
Z	1	?	0	0	0.000	-0.083	0.000	0.000	-0.071	NA

\end{table}

We see that nearly all of the subjects selected a response consistent with one of the identified interpretations. Responses that do not accord with any interpretation are indicated as `?` .

Note that options highlighted in light grey are considered within the range of 'visual error', defined by 0.5hr offset from the interpretation-specific projection.

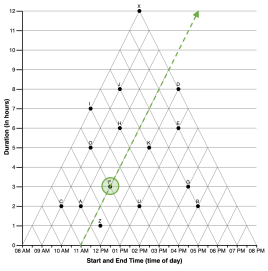
Which shifts start at 11am?



Response: A

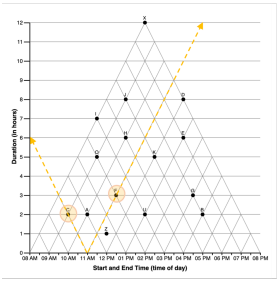
- indicates an **orthogonal** (incorrect) interpretation of the coordinate system
- Consistent with the reader identifying the reference point (11am) on the x-axis, *projecting an invisible orthogonal line upward*, and locating data point **A**.

Which shifts start at 11am?



Response: F

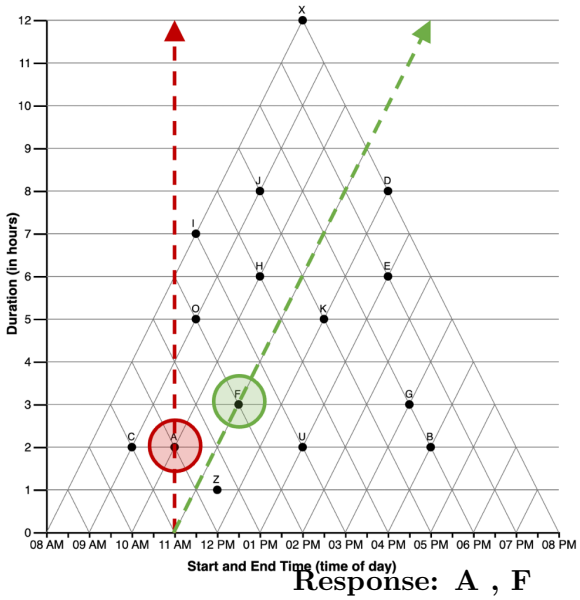
- indicates the **triangular** (correct) interpretation of the coordinate system
- Consistent with the reader identifying the reference point (11am) on the x-axis, and following the right-diagonal gridline, identifying data point **F**.



Response: C, F

- indicates a **maximal-Tversky** strategy following connecting lines
- Consistent with the reader identifying the reference point (11am) on the x-axis, and following *both* the right-diagonal and left-diagonal gridlines, identifying both datapoints F and C gridline.

Which shifts start at
11am?

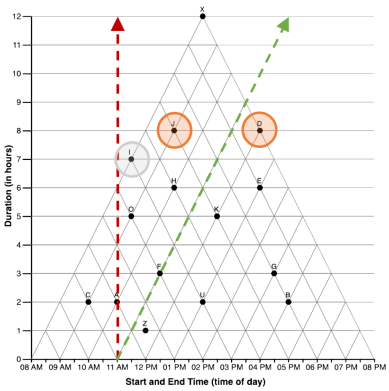


Response: A , F

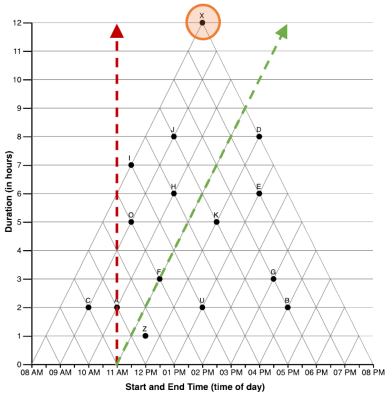
- The reader selects both triangular and orthogonal-consistent data points
- Possibly indicates uncertainty or confusion

A number of alternative responses were given:

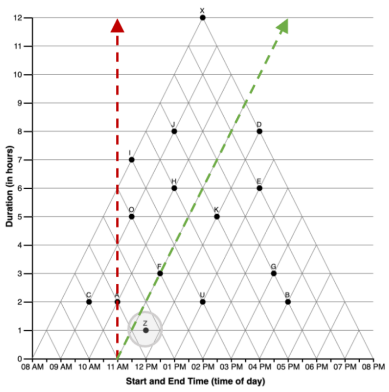
I J D



X



Z



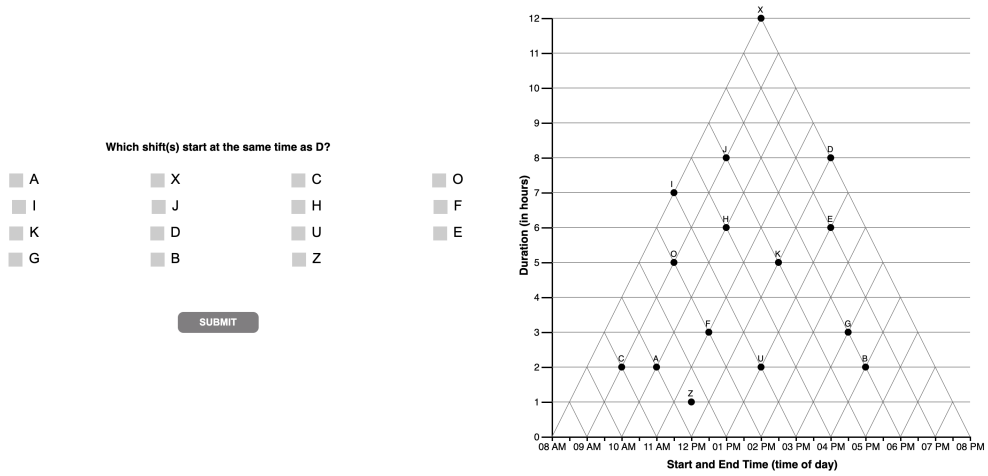


Figure 2.4: Q2—Control Condition

Table 2.7: Answer Key | Q2 Control Condition : Which shift(s) start at the same time as D?

interpretation	answer	not penalized
Triangular	K	Z
Orthgonal	E	G
Tversky [maximal]	AKJX	Z
Tversky [start diagonal]	AK	Z
Tversky [end diagonal]	X	
Tversky [duration line]	J	

Note: 15 response options: AIKGXJDBCHUZOFE

2.4.1.2 Question #2

```
q <- keys_raw %>% filter(condition == 111) %>% filter(Q==2)
ignore <- q %>% select("ALSO")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL","TV_max","TV_start", "TV_end", "TV_dur"
ves <- q %>% select("TRIve", "ORTHve","TV_max_ve","TV_start_ve","TV_end_ve", "TV_dur_ve")%
options <- q %>% select("OPTIONS")
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Tversky [maximal]", "Tversky [start diagonal]",
            "Tversky [end diagonal]", "Tversky [duration line]")
d = tibble(interpretation = scores, answer = answers, allowed=ves)
d$answer <- replace_na(d$answer, "")
d$allowed <- replace_na(d$allowed, "")

title = paste("Answer Key | Q2 Control Condition : ", question)
cols = c("interpretation", "answer","not penalized")

d %>% kbl(caption = title, col.names = cols) %>% kable_classic() %>%
  footnote(general = paste("15 response options: ", options), general_title = "Note: ",foo
```

```
names = c("response","n", "interpretation","strict","nice","tri","orth","tv_max","tv_start
title <- "Frequency of Selected Response Options for Question #2 (Control Condition)"

df_items %>%  mutate_if(is.numeric, format, digits=2,nsmall = 0) %>% filter(q == 2 & condi
 dplyr::summarise( count = n(),
                   strict = unique(score_ABS),
                   nice = unique(score_niceABS),
                   triangular = unique(score_TRI),
                   orthogonal =  unique(score_ORTH),
```

```
                    tversky_max = unique(score_TV_max),
                    tversky_start = unique(score_TV_start),
                    tversky_end = unique(score_TV_end),
                    tversky_duration = unique(score_TV_duration),
                    interpretation = unique(interpretation)) %>%
  arrange(interpretation, desc(count)) %>%
  select(response, count, interpretation, strict, nice, triangular, orthogonal,
         tversky_max, tversky_start, tversky_end, tversky_duration) %>%
  kable(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Scores" = 2, "Interpretation Scores"=6)) %>%
  pack_rows("Orthogonal", 1, 3) %>%
  pack_rows("Triangular", 4, 5) %>%
  pack_rows("Lines-Connect", 6, 7) %>%
  pack_rows("other", 8, 10)
```

\begin{table}

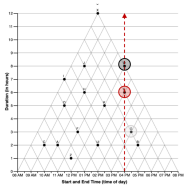
\caption{Frequency of Selected Response Options for Question #2 (Control Condition)}

			Strict Scores		Interpretation Scores					
response	n	interpretation	strict	nice	tri	orth	tv_max	tv_start	tv_end	tv_dur
Orthogonal										
E	121	Orthogonal	0	0	-0.083	1.000	-0.111	-0.091	-0.077	-0.077
DE	3	Orthogonal	0	0	-0.083	1.000	-0.111	-0.091	-0.077	-0.077
EG	1	Orthogonal	0	0	-0.167	1.000	-0.222	-0.182	-0.154	-0.154
Triangular										
K	24	Triangular	1	1	1.000	-0.083	0.250	0.500	-0.077	-0.077
KD	1	Triangular	0	1	1.000	-0.083	0.250	0.500	-0.077	-0.077
Lines-Connect										
J	4	Tversky	0	0	-0.083	-0.083	0.250	-0.091	-0.077	1.000
AK	1	Tversky	0	0	0.917	-0.167	0.500	1.000	-0.154	-0.154
other										
D	1	reference	0	0	0.000	0.000	0.000	0.000	0.000	0.000
B	1	?	0	0	-0.083	-0.083	-0.111	-0.091	-0.077	-0.077
C	1	?	0	0	-0.083	-0.083	-0.111	-0.091	-0.077	-0.077

\end{table}

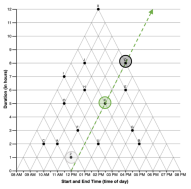
Again, we see that most subjects selected a response consistent with one of the identified interpretations. (note, when the question stem includes a data point rather than time as reference, we do not penalize respondents for selecting the reference data point *in addition* to an interpretation consistent response. For example, in this question, we do not penalize respondents for selecting option D, the reference point in the question.)

Which shift(s) start at the same time as D?



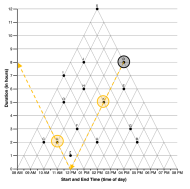
Response: E (also EG, DE)

- indicates an **orthogonal** (incorrect) interpretation of the coordinate system
- Consistent with the reader identifying the reference point (D) on the graph, *projecting an invisible orthogonal line through it*, and locating data point **E**.



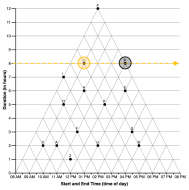
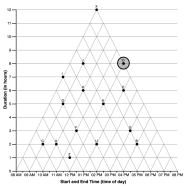
Response: K (also KD)

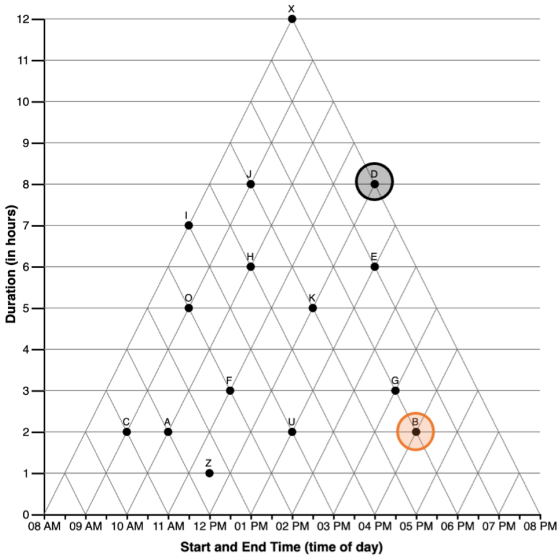
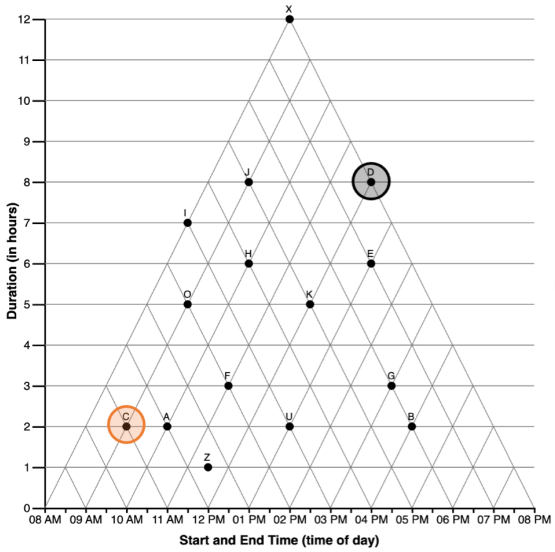
- indicates an **triangular** (correct) interpretation of the coordinate system
- Consistent with the reader identifying the reference point (D) on the graph, and following its *descending-leftward diagonal gridline*, and locating data point **K**.



Response: AK

- indicates an **Tversky** strategy following connecting lines
- Consistent with the reader identifying the reference point (D) on the graph, and following its *descending-leftward diagonal gridline*, and locating data point **K** then *continuing along the connecting ascending leftward diagonal* locating data point A.

Which shift(s) start at the same time as D?	
	**Response: J** • indicates an **Tversky** strategy following connecting lines • Consistent with the reader identifying the reference point (D) on the graph, and following its horizontal gridline to the y-axis, locating data point J.
	**Response: D** • the reader selected only the **reference point** • Consistent with the reader identifying the reference point (D) on the graph • Possibly indicates uncertainty or confusion

B	C
	

2.4.1.3 Question #3

```
q <- keys_raw %>% filter(condition == 111) %>% filter(Q==3)
ignore <- q %>% select("ALSO")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL","TV_max","TV_start", "TV_end", "TV_dur"
ves <- q %>% select("TRIve", "ORTHve","TV_max_ve","TV_start_ve","TV_end_ve", "TV_dur_ve")%
options <- q %>% select("OPTIONS")
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Tversky [maximal]", "Tversky [start diagonal]",
            "Tversky [end diagonal]", "Tversky [duration line]")
d = tibble(interpretation = scores, answer = answers, allowed=ves)
d$answer <- replace_na(d$answer, "")
d$allowed <- replace_na(d$allowed, "")

title = paste("Answer Key | Q2 Control Condition : ", question)
cols = c("interpretation", "answer","not penalized")

d %>% kbl(caption = title, col.names = cols) %>% kable_classic() %>%
  footnote(general = paste("15 response options: ", options), general_title = "Note: ",foo
```

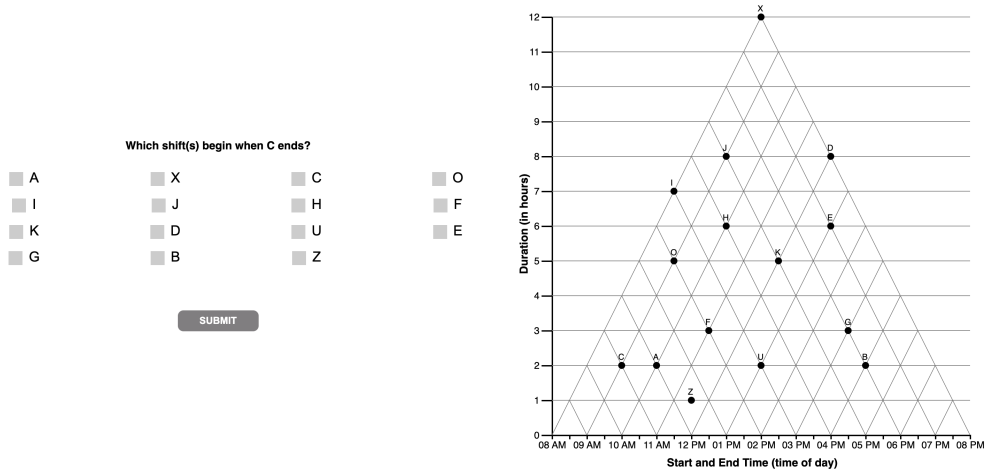


Figure 2.5: Q3—Control Condition

Table 2.10: Answer Key | Q2 Control Condition : Which shift(s) begin when C ends?

interpretation	answer	not penalized
Triangular	F	Z
Orthgonal	Z	FIO
Tversky [maximal]	AUBFOJ	
Tversky [start diagonal]	OJ	
Tversky [end diagonal]	F	Z
Tversky [duration line]	AUB	

Note: 15 response options: AIKGXJDBCHUZOFE

```
names = c("response","n","interpretation",
          "strict","nice","tri","orth","tv_max","tv_start","tv_end","tv_dur")
title <- "Frequency of Selected Response Options for Question #3 (Control Condition)"

df_items %>% filter(q == 3 & condition == 111) %>% group_by(response) %>%
   dplyr::summarise( count = n(),
                     strict = unique(score_ABS),
                     nice = unique(score_niceABS),
                     triangular = unique(score_TRI),
                     orthogonal =  unique(score_ORTH),
                     tversky_max = unique(score_TV_max),
                     tversky_start = unique(score_TV_start),
                     tversky_end = unique(score_TV_end),
                     tversky_duration = unique(score_TV_duration),
                     interpretation = unique(interpretation)) %>%
  arrange(interpretation, desc(count)) %>%
  select(response, count, interpretation, strict, nice, triangular, orthogonal,
         tversky_max, tversky_start, tversky_end, tversky_duration) %>%
  kbl(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Scores" = 2, "Interpretation Scores"=6)) %>%
  pack_rows("Orthogonal", 1, 1) %>%
  pack_rows("Triangular", 2, 3) %>%
  pack_rows("Lines-Connect", 4, 8) %>%
  pack_rows("other", 9, 17)
```

\begin{table}

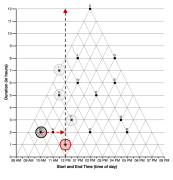
\caption{Frequency of Selected Response Options for Question #3 (Control Condition)}

			Strict Scores		Interpretation Scores				
response	n	interpretation	strict	nice	tri	orth	tv_max	tv_start	tv_end
Orthogonal									
Z	94	Orthogonal	0	0	0.000	1.0	-0.125	-0.083	0.000
Triangular									
F	24	Triangular	1	1	1.000	0.0	0.167	-0.083	1.000
FKE	1	Triangular	0	0	0.833	-0.2	-0.083	-0.250	0.833
Lines-Connect									
AUB	4	Tversky	0	0	-0.250	-0.3	0.500	-0.250	-0.250
O	3	Tversky	0	0	-0.083	0.0	0.167	0.500	-0.083
OJ	2	Tversky	0	0	-0.167	-0.1	0.333	1.000	-0.167
OJD	1	Tversky	0	0	-0.250	-0.2	0.208	0.917	-0.250
OK	1	Tversky	0	0	-0.167	-0.1	0.042	0.417	-0.167
other									
C	1	reference	0	0	0.000	0.0	0.000	0.000	0.000
AIOZFHJXKUDEGB	1	frenzy	0	0	0.000	0.0	0.000	0.000	0.000
A	18	?	0	0	-0.083	-0.1	0.167	-0.083	-0.083
K	3	?	0	0	-0.083	-0.1	-0.125	-0.083	-0.083
AH	1	?	0	0	-0.167	-0.2	0.042	-0.167	-0.167
DE	1	?	0	0	-0.167	-0.2	-0.250	-0.167	-0.167
E	1	?	0	0	-0.083	-0.1	-0.125	-0.083	-0.083
U	1	?	0	0	-0.083	-0.1	0.167	-0.083	-0.083
UE	1	?	0	0	-0.167	-0.2	0.042	-0.167	-0.167

\end{table}

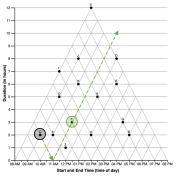
TODO address RESPONSE FKE which is classified as Triangular but doesn't seem to fit this interpretation

What shift(s) begin when C ends?



Response: A

- indicates an **orthogonal** (incorrect) interpretation of the coordinate system
- Consistent with the reader identifying the reference point (11am) on the graph, and using the duration encoded on the y-axis (2) , project along the horizontal gridline by two hours, and then *project an invisible orthogonal line through that time (12PM)* locating data point **Z**.



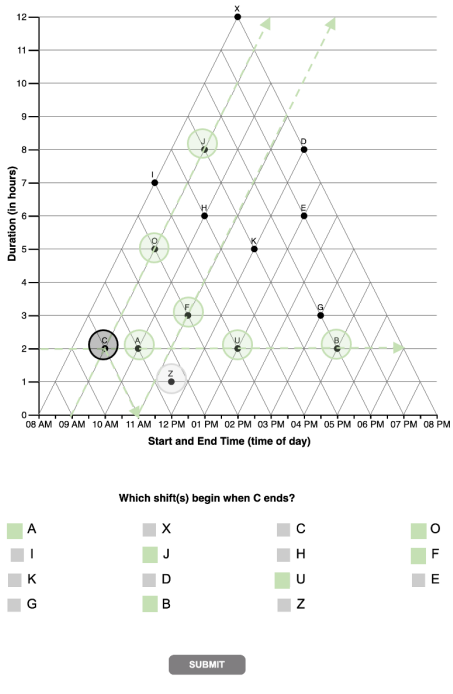
Response: F

- indicates a (correct) **triangular** interpretation of the coordinate system
- Consistent with the reader identifying the reference point (11am) on the graph, and following the *descending gridline* to the x-axis to identify the end-time (11AM) and then following the *ascending gridline* to identify datapoints starting at 11AM and locating data point **F**.

Correct — Triangular

Incorrect — Lines Connect

Incorrect — Orthogonal



A correct (triangular) interpretation of the coordinate system yields data point K.

An incorrect interpretation that is not orthogonal yields selection of datapoints that are connected by a line to data point D, such as K and A, or alternatively, J.

An incorrect orthogonal interpretation involves selecting data point E, the orthogonal projection from the reference point D to the x-axis.

2.4.1.4 Question #4

```
#define interpretation for each response
#NOTE: this is manually defined, based on inspecting answer options
  interpretation = c("ORTHOGONAL","TRIANGULAR","orth-lines connect","BLANK","orth-lines co
                     "lines-connect","orthogonal","","","",
                     "","","","","","" )

title <- "Frequency of Selected Response Options for Question #4 (Control Condition)"
df_items %>% filter(q == 4 & condition == 111) %>% group_by(response) %>%  dplyr::summaris
                    strict = unique(score_ABS),
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
```

```
                    orthogonal =  unique(score_ORTH),
                    DISC =  triangular - orthogonal
                    # DISC = (score_TRI - score_ORTH)
                    ) %>% arrange(-count) %>% cbind(interpretation) %>%
  arrange(desc(interpretation)) %>%
  select(response, count, interpretation, strict, nice, triangular, orthogonal, DISC) %>%
  kbl(caption = title) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Scores" = 2, "Interpretation Scores"=2, " "=1)) %>%
  pack_rows("Triangular", 1, 1) %>%
  pack_rows("Orthogonal", 2, 5) %>%
  pack_rows("Lines-Connect", 6, 6) %>%
  pack_rows("other", 7, 16)
```

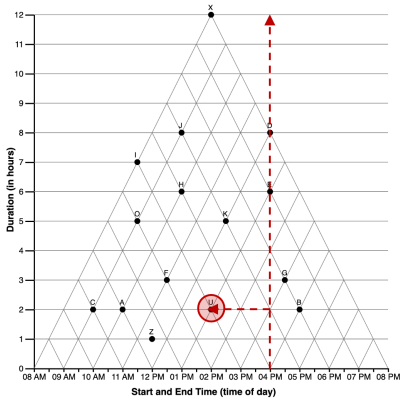
\begin{table}

\caption{Frequency of Selected Response Options for Question #4 (Control Condition)}

			Strict Scores		Interpretation Scores		
response	count	interpretation	strict	nice	triangular	orthogonal	DISC
Triangular							
H	29	TRIANGULAR	1	1	1.000	-0.071	1.07
Orthogonal							
U	87	ORTHOGONAL	0	0	-0.071	1.000	-1.07
FU	2	orthogonal	0	0	-0.143	0.929	-1.07
DE	14	orth-lines connect	0	0	-0.143	-0.143	0.00
E	6	orth-lines connect	0	0	-0.071	-0.071	0.00
Lines-Connect							
B	3	lines-connect	0	0	-0.071	-0.071	0.00
other							
	6	BLANK	0	0	0.000	0.000	0.00
K	2		0	0	-0.071	-0.071	0.00
O	2		0	0	-0.071	-0.071	0.00
AH	1		0	0	0.929	-0.143	1.07
CAIOZFHJXKU	1		0	0	0.286	0.286	0.00
D	1		0	0	-0.071	-0.071	0.00
G	1		0	0	-0.071	-0.071	0.00
KU	1		0	0	-0.143	0.929	-1.07
OUDE	1		0	0	-0.286	0.786	-1.07
UDE	1		0	0	-0.214	0.857	-1.07

\end{table}

In Question 4 we see more than one variant of an orthogonal interpretation emerge.

Orthogonal	Orthogonal-LinesConnecting
	
If the subject calculates end time for each data point (using duration on the y axis), they find that an (incorrect) projection of point U 'end time' intersects with the (incorrect) orthogonal projection of 4:00PM.	Alternatively, some subjects selected points E and D which intersect with an orthogonal projection from 4:00pm. We call this an 'orthogonal-lines connect" strategy, because it (incorrectly) adapts the orthogonal procedure for finding events that *start* at 4:00pm in order to find those that *end* at 4:00pm, thus selecting any data point with an orthogonal intersection with 4:00pm.

2.4.1.5 Question #5

```
#define interpretation for each response
#NOTE: this is manually defined, based on inspecting answer options
  interpretation = c("ORTHOGONAL","TRIANGULAR","orth-satistice","BLANK","orth-satisfice",
                     "orth-satistice","","","","","orth-satisfice","","orth-satisfice","",
                     "(reference point)","","","","lines-connect")

title <- "Frequency of Selected Response Options for Question #4 (Control Condition)"
df_items %>% filter(q == 5 & condition == 111) %>% group_by(response) %>%
  dplyr::summarise( count = n(),
                    strict = unique(score_ABS),
```

```
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
                    orthogonal =  unique(score_ORTH),
                    DISC =  triangular - orthogonal
                    # DISC = (score_TRI - score_ORTH)
                    ) %>% arrange(-count) %>% cbind(interpretation) %>%
  arrange(desc(interpretation)) %>%
  select(response, count, interpretation, strict, nice, triangular, orthogonal, DISC) %>%
  kbl(caption = title) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Scores" = 2, "Interpretation Scores"=2, " "=1)) %>%
  pack_rows("Triangular", 1, 1) %>%
  pack_rows("Orthogonal", 2, 9) %>%
  pack_rows("Lines Connect", 10, 10) %>%
  pack_rows("other", 11, 22)
```

\begin{table}

\caption{Frequency of Selected Response Options for Question #4 (Control Condition)}

			Strict Scores		Interpretation Scores		
response	count	interpretation	strict	nice	triangular	orthogonal	DISC
Triangular							
O	50	TRIANGULAR	1	1	1.000	-0.077	1.077
Orthogonal							
U	64	ORTHOGONAL	0	0	-0.091	1.000	-1.091
F	10	orth-satistice	0	0	-0.091	-0.077	-0.014
J	3	orth-satistice	0	0	-0.091	-0.077	-0.014
H	3	orth-satisfice	0	0	-0.091	-0.077	-0.014
K	2	orth-satisfice	0	0	-0.091	-0.077	-0.014
FK	1	orth-satisfice	0	0	-0.182	-0.154	-0.028
HJ	1	orth-satisfice	0	0	-0.182	-0.154	-0.028
HU	1	orth-satisfice	0	0	-0.182	0.923	-1.105
Lines Connect							
Z	1	lines-connect	0	0	0.000	-0.077	0.077
other							
	6	BLANK	0	0	0.000	0.000	0.000
I	1	(reference point)	0	0	0.000	0.000	0.000
OF	3		0	0	0.909	-0.154	1.063
B	2		0	0	-0.091	-0.077	-0.014
FG	2		0	0	-0.182	-0.154	-0.028
JD	2		0	0	-0.182	-0.154	-0.028
C	1		0	0	-0.091	-0.077	-0.014
G	1		0	0	-0.091	-0.077	-0.014
HJDE	1		0	0	-0.364	-0.308	-0.056
OH	1		0	0	0.909	-0.154	1.063
OK	1		0	0	0.909	-0.154	1.063
X	1		0	0	-0.091	-0.077	-0.014

\end{table}

TODO note the compelling cases of internal inconsistency (HJDE)

2.4.2 Impasse Condition

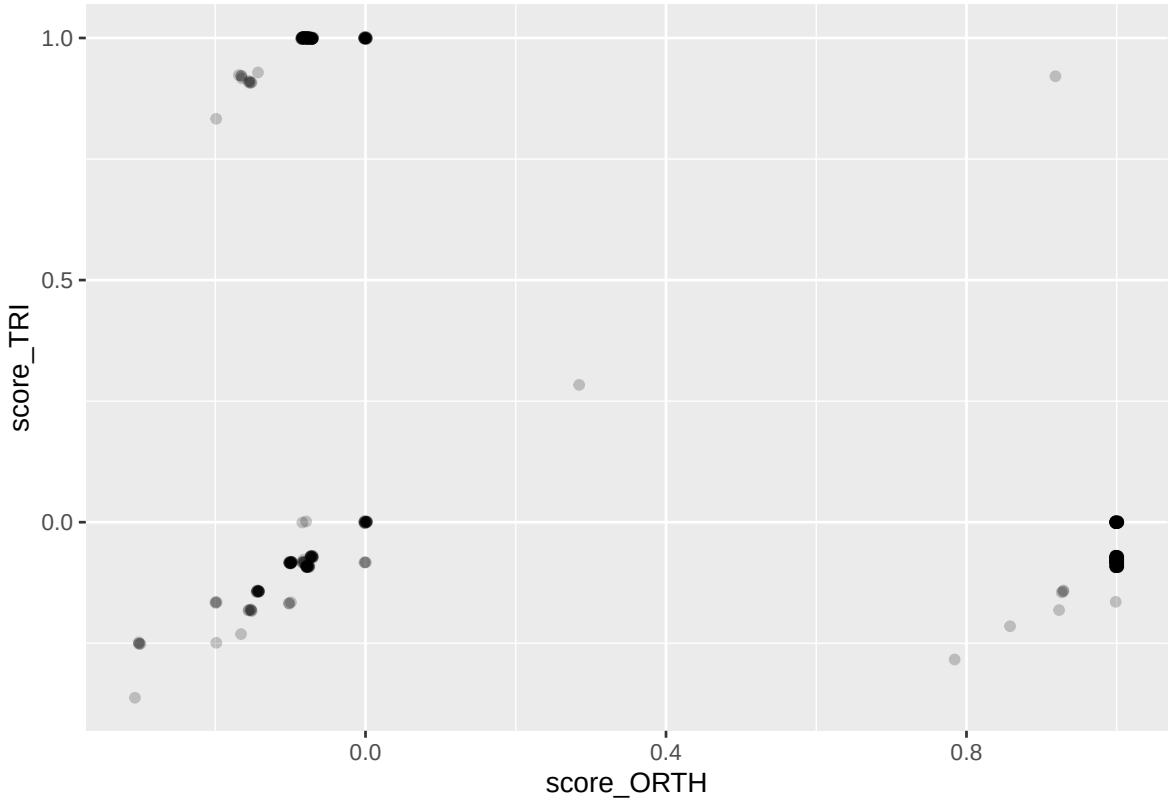
We can compare these responses to those produced by participants in the experimental impasse condition (`condition` = 121).

```
#
# #define interpretation for each response
# #NOTE: this is manually defined, based on inspecting answer options
```

```
# interpretation = c("BLANK","TRIANGULAR","partial-satisfice","LINES-CONNECT",
#                    "partial-satisfice","partial-satisfice","partial-lines-connect",
#                    "","partial-satisfice","partial-satisfice","partial-satisfice","parti
#
# title <- "Frequency of Selected Response Options for Question #1 (Impasse Condition)"
# df_items %>% filter(q == 1 & condition == 121) %>% group_by(response) %>%
#    dplyr::summarise( count = n(),
#                      strict = unique(score_ABS),
#                      nice = unique(score_niceABS),
#                      triangular = unique(score_TRI),
#                      orthogonal =  unique(score_ORTH),
#                      DISC =  triangular - orthogonal
#                      # DISC = (score_TRI - score_ORTH)
#                      ) %>% arrange(-count) %>% cbind(interpretation) %>%
#   arrange(desc(interpretation)) %>%
#   select(response, count, interpretation, strict, nice, triangular, orthogonal, DISC) %>
#   kbl(caption = title) %>%  kable_classic() %>%
#   add_header_above(c(" " = 3, "Strict Scores" = 2, "Interpretation Scores"=2, " "=1))
```

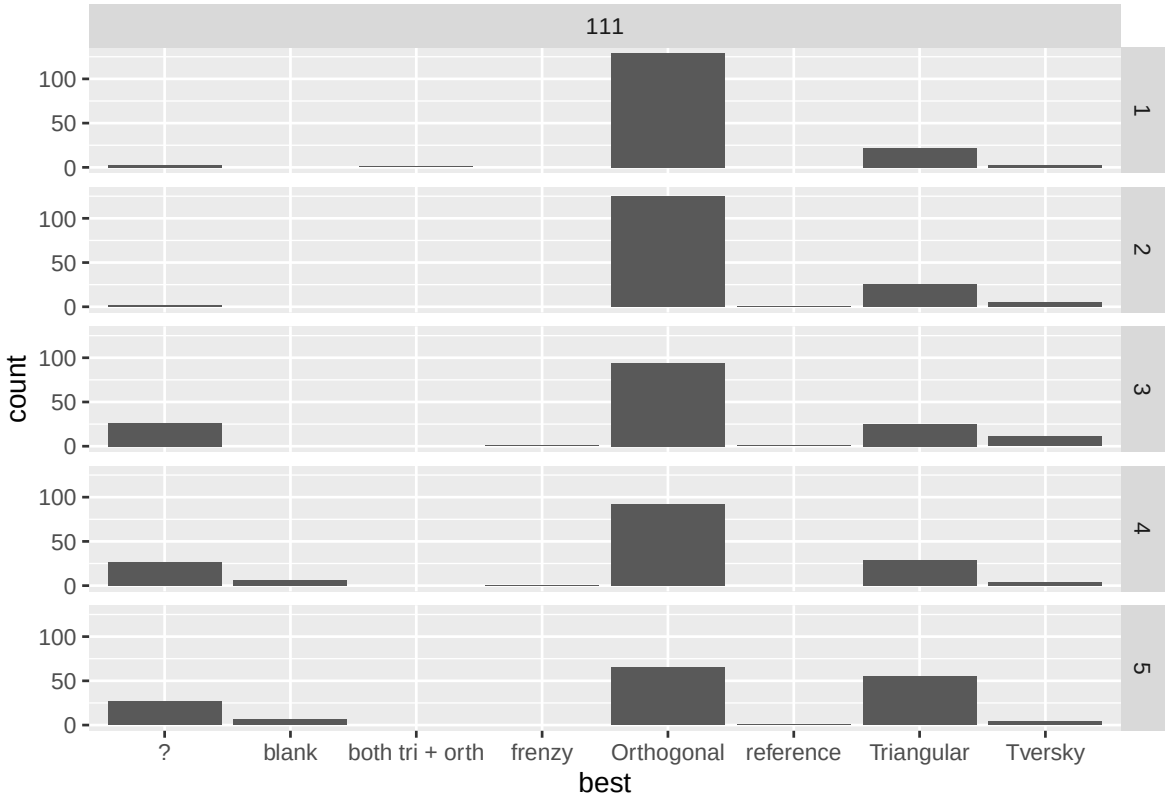
2.4.3 Summarize By subject

```
# df_items$DISC <- df_items$score_TRI - df_items$score_ORTH
gf_jitter(score_TRI ~ score_ORTH, data = df_items, alpha = 0.2)
```




```
# gf_jitter(score_TRI ~ DISC, data = df_items , alpha=0.2,  color = ~q)
# gf_dhistogram(~df_items$DISC, data = df_items)

gf_bar(~best, data = df_items) %>% gf_facet_grid(q~condition)
```




```
subject_scores <- df_items %>% group_by(subject) %>% dplyr::summarize(
  TRI = sum(score_TRI),
  ORTH = sum(score_ORTH),
  # DISCRIM = sum(score_TRI + score_ORTH),
  ABSOLUTE = sum(correct),
  condition = unique(condition)
)
```

```
`summarise()` ungrouping output (override with `.groups` argument)
```

```
head(subject_scores)
```

```
# gf_dhistogram(~DISCRIM, data = subject_scores)  %>%  gf_facet_grid(~condition)
#
# gf_dhistogram(~ABSOLUTE, data = subject_scores)
#
```

subject	TRI	ORTH	ABSOLUTE	condition
04YXZ	0.356	1.4	0	111
0F1A8	-0.323	3.92	0	111
194UU	-0.573	3.7	0	111
19SNU	-0.323	5	0	111
1ZY3L	4.92	0.685	4	111
2BBZ9	-0.323	5	0	111

```
# gf_jitter(ABSOLUTE ~ DISCRIM, data = subject_scores, alpha = 0.2)
```

TODO TRAPDOOR?

SET T = options that result in -1 score (trapdoor)

TODO: TEST first with NO trapdoor, and see what the scores yield, then if the responses can be scaled based on comparing subscores

f_partialP <- function(t,p,f,q) {

t = number of correct-selected options p = number of true options f = number of incorrect-selected options q = number of false options n = number of options + p + q return((t / p) - (f/q)) }

To prepare for partial scoring [-1/q, + 1/p], we first need to define the parameters t, p , f, q

Our approach to calculating an ORTHOGONAL PARTIAL score is as follows:

- 1/p for each correct-selected
- -1/q for each incorrect-selected (except 'allowed' based on reference point and visual error)
- trapdoor : default to -1 if select triangular

3 RESOURCES

set operations https://stat.ethz.ch/R-manual/R-devel/library/base/html/sets.html

4 Exploration

THIS NOTEBOOK IS INCOMPLETE

TODO

- double check which Qs in task are non-discriminating (6 and 9?)

```
library(tidyverse) #ALL THE THINGS
library(kableExtra) #printing tables
library(mosaic) #simple descriptives [favstats]
library(Hmisc) # %nin% operator


library(ggpubr) #arrange plots

# library(pastecs) #stat.desc
# library(car) #ANOVA, qqplot
# library(effectsize) #effect size

#set some output options
library(dplyr, warn.conflicts = FALSE)
options(dplyr.summarise.inform = FALSE)
options(ggplot2.summarise.inform = FALSE)
options(scipen=1, digits=3)
```

The purpose of this notebook is explore the distribution of dependent variables for Study SGC3A.

Pre-Requisite	Followed By
1_sgc3A_harmonize.qmd	
2_sgc3A_rescoring.qmd	

```
#IMPORT DATA
df_items <- read_rds('data/sgc3a_items.rds')
df_subjects <- read_rds('data/sgc3a_participants.rds')
```

Table 4.2: Participants by Condition and Data Collection Period

	Control Condition	Impasse Condition	Total for Period
winter22	28	37	65
fall17	27	27	54
spring18	35	37	72
fall21	68	71	139
Sum	158	172	330

```
#SEPARATE ITEM DATA BY QUESTION TYPE
df_scaffold <- df_items %>% filter(q < 6)
df_test <- df_items %>% filter(q > 6) %>% filter (q %nin% c(6,9))
```

4.1 Sample

4.1.1 Data Collection

Data was initially collected (in person, SONA groups in computer lab) in Fall 2017. In Spring 2018, additional data were collected after small modifications were made to the experimental platform to increase the size of multiple-choice input buttons, and to add an additional free-response question following the main task block. In Fall 2021, the study was replicated using asynchronous, online SONA pool, with additional participants collected in Winter 2022.

```
title = "Participants by Condition and Data Collection Period"
cols = c("Control Condition","Impasse Condition","Total for Period")
cont <- table(df_subjects$term, df_subjects$condition)
cont %>% addmargins() %>% kbl(caption = title, col.names = cols) %>%  kable_classic()
```

4.1.2 Participants

```
#Describe participants
subject.stats <- rbind(
  "lab"= df_subjects %>% filter(mode == 'lab-synch') %>% select(age) %>% unlist() %>% favs
  "online" = df_subjects %>% filter(mode == "asynch") %>% select(age) %>% unlist() %>% fav
)
subject.stats$female <- c(
```

Table 4.3: Descriptive Statistics of Participant Age and Gender

	min	Q1	median	Q3	max	mean	sd	n	missing	female
lab	18	19	20	21	33	20.4	2.12	126	0	78
online	18	20	20	21	31	20.6	2.00	204	0	137

```
  (df_subjects %>% filter(mode == 'lab-synch') %>%  filter(gender=="Female") %>% count())$
  (df_subjects %>% filter(mode == "asynch") %>% filter(gender=="Female") %>% count())$n
)

title = "Descriptive Statistics of Participant Age and Gender"
subject.stats %>% kbl (caption = title) %>% kable_classic()
```

For **in-person** collection, 126 participants (60 % female) undergraduate STEM majors at a public American University participated *in person* in exchange for course credit (age: 18 - 33 years). Participants were randomly assigned to one of two experimental groups.

For **online replication** 204 participants (70 % female) undergraduate STEM majors at a public American University participated *online, asynchronously* in exchange for course credit (age: 18 - 31 years). Participants were randomly assigned to one of two experimental groups.

4.2 Response Accuracy

4.2.1 Cumulative Scores

Cumulative scores indicate the response accuracy by participant across all items in the graph comprehension task.

4.2.1.1 Cumulative Absolute Score

Recall from Section 2.1.2.1 that the absolute score (following the dichotomous scoring approach) indicates if the subject's response for a particular item was *perfectly* correct: whether they selected all correct answer options and no others. The absolute score for an individual item is either 0 or 1. When cumulated across the entire set of items, the cumulative absolute score for an individual subject ranges from [0,15].

```
title = "Descriptive Statistics of Response Accuracy (Absolute Score)"
abs.stats <- rbind(
  "lab"= df_subjects %>% filter(mode == 'lab-synch') %>% select(absolute_score) %>% unlist
```

Table 4.4: Descriptive Statistics of Response Accuracy (Absolute Score)

	min	Q1	median	Q3	max	mean	sd	n	missing
lab	1	2	3	11	15	5.81	4.89	126	0
online	0	2	2	8	15	4.64	4.73	204	0

```
  "online" = df_subjects %>% filter(mode == "asynch") %>% select(absolute_score) %>% unlis
)
abs.stats %>% kbl (caption = title) %>% kable_classic()
```

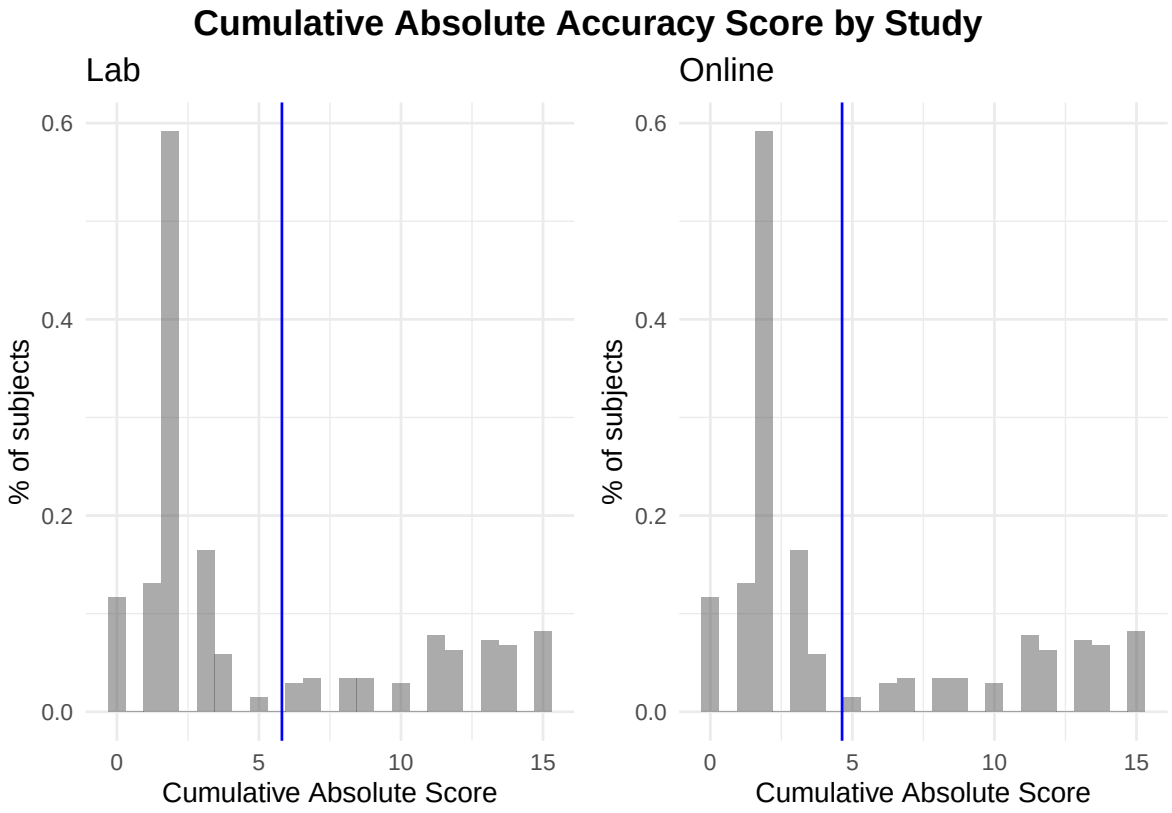
For *in person* collection, cumulative absolute scores (n = 126) range from 1 to 15 with a mean score of (M = 5.81, SD = 4.89).

For *online replication*, (online) cumulative accuracy scores (n = 204) range from 0 to 15 with a mean score of (M = 4.64, SD = 4.73).

```
#VISUALIZE distribution of response accuracy
plab <- gf_dhistogram(~absolute_score, data = df_subjects) %>%
  gf_vline(xintercept = ~abs.stats["lab",]$mean, color = "blue") +
  labs(title="Lab", x = "Cumulative Absolute Score", y="% of subjects") + theme_minimal()

ponline <- gf_dhistogram(~absolute_score, data = df_subjects) %>%
  gf_vline(xintercept = ~abs.stats["online",]$mean, color = "blue") +
  labs(title="Online", x = "Cumulative Absolute Score", y="% of subjects") + theme_minimal

plot <-ggarrange(plab, ponline, common.legend = TRUE, nrow = 1, ncol =2)
annotate_figure(plot, top = text_grob("Cumulative Absolute Accuracy Score by Study",
   color = "black", face = "bold", size = 14))
```



TODO double check that the subject-totals match the sum of the item level totals

4.2.1.2 Cumulative Interpretation Scores

4.2.2 Item Score

Item scores indicate the response accuracy by a participant on each individual question in the graph comprehension task.

4.2.2.1 Item Absolute Score

4.2.2.2 Item Interpretation Scores

4.3 Response Latency

- TODO: Investigate super high and super low response times..

Table 4.5: Descriptive Statistics of Response Latency (Time on Study)

	min	Q1	median	Q3	max	mean	sd	n	missing
lab	6.01	10.50	12.2	14.4	23.9	12.8	3.37	126	0
online	2.91	9.18	11.5	15.0	111.0	13.4	9.21	204	0

- TODO: Investigate appropriate models for response time data. (see: https://lindeloev.github.io/shiny-rt/).
- Especially see https://lindeloev.github.io/shiny-rt/ for ideas on modelling reaction time data

4.3.1 Time on Study

```
#DESCRIBE distribution of response time
time.stats <- rbind(
  "lab"= df_subjects %>% filter(mode == 'lab-synch') %>% select(totaltime_m) %>% unlist()
  "online"= df_subjects %>% filter(mode == 'asynch') %>% select(totaltime_m) %>% unlist()
)

title = "Descriptive Statistics of Response Latency (Time on Study)"
time.stats %>% kbl(caption = title) %>% kable_classic()
```

Total time on study for *in person* subjects (n = 126) ranged from 6.01 to 23.86 minutes with a mean duration of (M = 12.8, SD = 3.37).

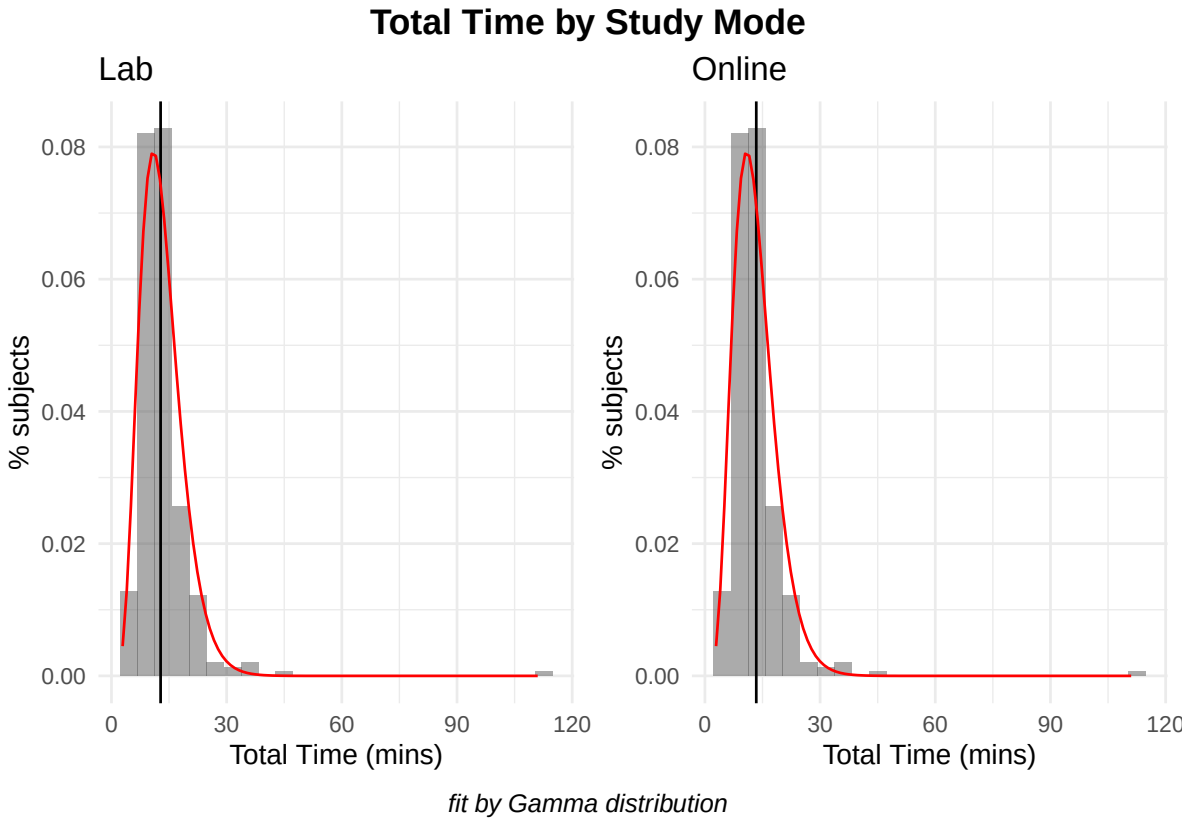
Total time on study for *online replication* subjects (n = 204) ranged from 2.91 to 111.02 minutes with a mean duration of (M = 13.37, SD = 9.21).

```
#VISUALIZE distribution of response time
plab <- gf_dhistogram(~totaltime_m, data = df_subjects) %>%
  gf_vline(xintercept = ~time.stats["lab",]$mean, color = "black") %>%
  gf_fitdistr(dist="gamma", color="red")+
  labs(title="Lab", x = "Total Time (mins)", y = "% subjects") +  theme_minimal()

ponline <- gf_dhistogram(~totaltime_m, data = df_subjects) %>%
  gf_vline(xintercept = ~time.stats["online",]$mean, color = "black") %>%
  gf_fitdistr(dist = "gamma", color="red")+
  labs(title="Online", x = "Total Time (mins)", y = "% subjects") +  theme_minimal()

plot <-ggarrange(plab, ponline, common.legend = TRUE, nrow = 1, ncol =2)
```

```
annotate_figure(plot,
                top = text_grob("Total Time by Study Mode",color = "black", face = "bold",
                bottom = text_grob("fit by Gamma distribution", face = "italic", size = 10
```



TODO consider log transform of response latency data *see* archive sgc3A_participants.Rmd

4.3.2 Time on Question

TODO time on question

4.4 Resources

- https://rpkgs.datanovia.com/ggpubr/reference/index.html

References

Schmidt, Dennis, Tobias Raupach, Annette Wiegand, Manfred Herrmann, and Philipp Kanzow. 2021. "Relation Between Examinees' True Knowledge and Examination Scores: Systematic Review and Exemplary Calculations on Multiple-True-False Items." *Educational Research Review* 34 (November): 100409. https://doi.org/10.1016/j.edurev.2021.100409.