SGC3A

Study SGC3A | 5 Exploratory Analyses

Amy Rae Fox

4/29/2022

Table of contents

Status

- 6/27 | SGC3A refactor file names/refs to differentiate hypo testing from exploratory modelling
- 6/27 | SGC3A chisqr on first question
- 6/27 | add sample logistic regression learning file
- … restructure and creation of modelling file
- 5/24 | SGC3A SCORING| con't response exploration + add Q6-Q15 keys
- 5/24 | SGC3A EXPLORATION| quick and dirty lms given scaled_score at subject level
- 5/18 | SG3A SCORING | refactor scoring to differentiate SATISFICING left vs right
- 5/18 | SG3A SCORING | refactor scoring for performance (60mins / dataset to ~25 mins / dataset)
- 5/13 - 5/16 | SG3A SCORING | refactor code for robust to condition
- 5/11/22 | SGC3A SCORING | cleanup text + continue response strategy analysis 111=Q3
- 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

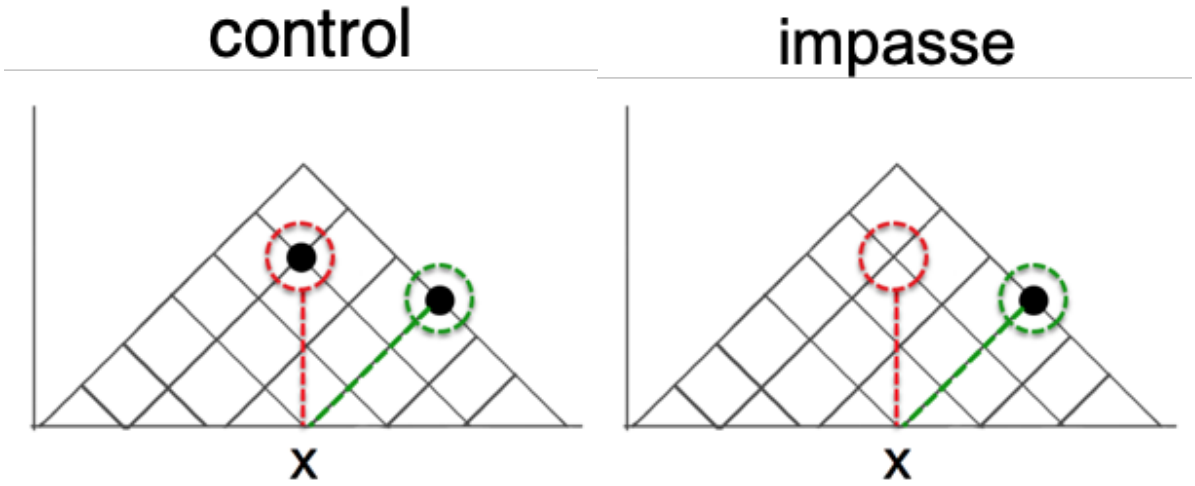
SGC3A

INTRODUCTION

TODO - add subsequent hypotheses and exploratory questions - placeholder hypothesis testing - placeholder exploratory analyses

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 **??**. 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.

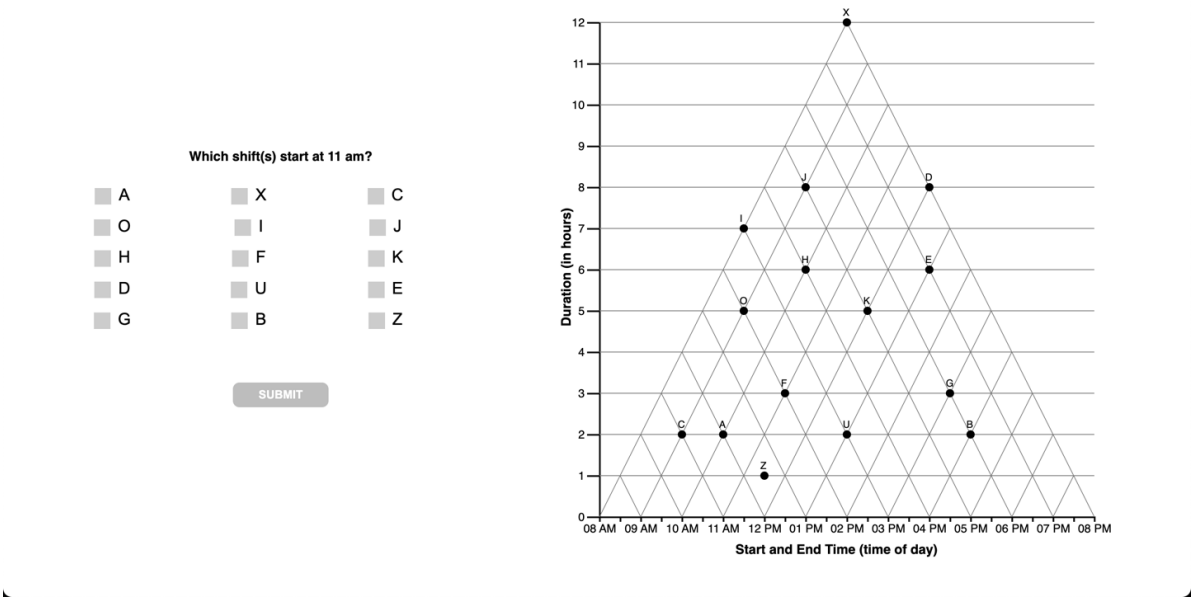


Figure 2: Sample Question (Q=1) for Graph Comprehension Task

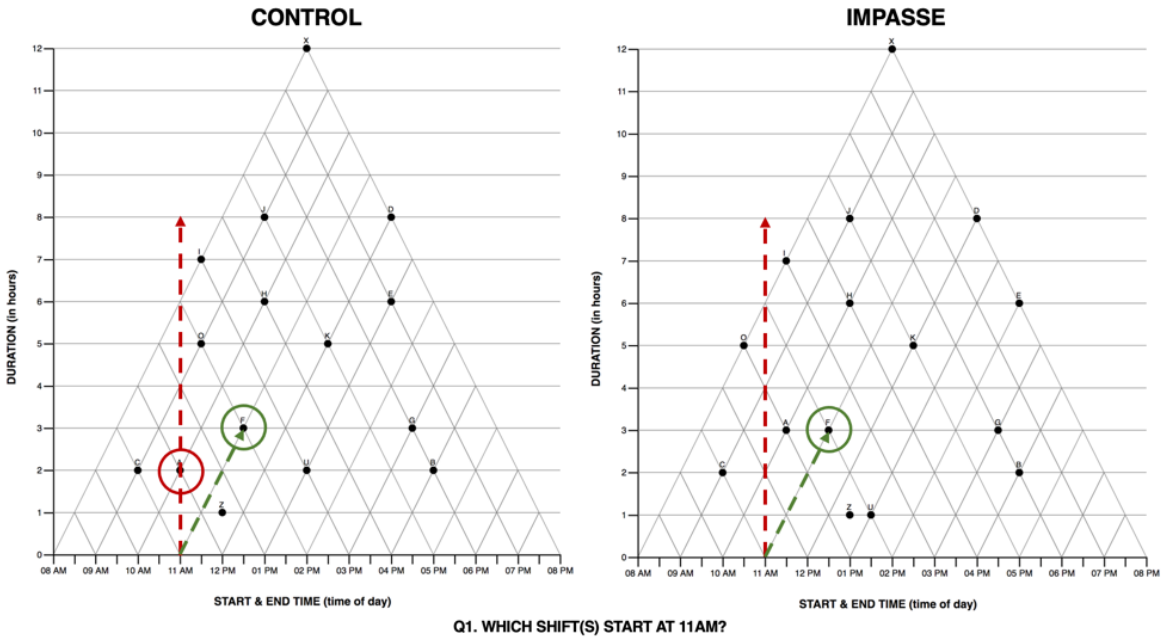


Figure 3: Sample Question (Q=1) graphs for each condition

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 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 **??**).

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 **??**.

Hypothesis Testing

ToDo

Exploratory Analysis

TODO

Discussion

TODO

1 Harmonization

TODO

- refactor code chunk labels

The purpose of this notebook is to harmonize data files collected across different data collection sessions 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_scoring.qmd

```
library(tidyverse) #ALL THE THINGS
library(codebook) #data dictionary

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

1.1 HARMONIZATION STRATEGY

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/0-session-level/fall17_sgc3a_participants.csv"
spring18 <- "data/0-session-level/spring18_sgc3a_participants.csv"
fall21 <- "data/0-session-level/fall21_sgc3a_participants.csv"
winter22 <- "data/0-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, levels= c("fall17","spring18","fall21","winter22")),
    mode = factor(mode),
    gender = factor(gender),
    schoolyear = factor(schoolyear, levels=c("First","Second","Third","Fourth","Fifth","Ot
  )

#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.

```
#set datafiles
fall17 <- "data/0-session-level/fall17_sgc3a_blocks.csv"
spring18 <- "data/0-session-level/spring18_sgc3a_blocks.csv"
fall21 <- "data/0-session-level/fall21_sgc3a_blocks.csv"
winter22 <- "data/0-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)
```

1.1.3 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/1-study-level/sgc3a_participants.csv", row.names = FALSE)
write.csv(df_items,"data/1-study-level/sgc3a_items.csv", row.names = FALSE)
```

```r
write.csv(df_freeresponse,"data/1-study-level/sgc3a_items.csv", row.names = FALSE)

#SAVE R Data Structures
#export R DATA STRUCTURES (include codebook metadata)
rio::export(df_subjects, "data/1-study-level/sgc3a_participants.rds") # to R data structur
rio::export(df_items, "data/1-study-level/sgc3a_items.rds") # to R data structure file
```

1.3 RESOURCES

```r
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

TODO

- mine for inline TODOs
- finish item level response exploration
- complete raw key for Q 6 - Q 15

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

1_sgc3A_harmonize.qmd

```
options(scipen=1, digits=3)

library(tidyverse) #ALL THE THINGS
library(kableExtra) #printing tables
library(ggformula) #quick graphs
library(ggpubr) #better graphs
library(pbapply) #progress bar and time estimate for *apply fns
library(Hmisc) # %nin% operator
```

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.

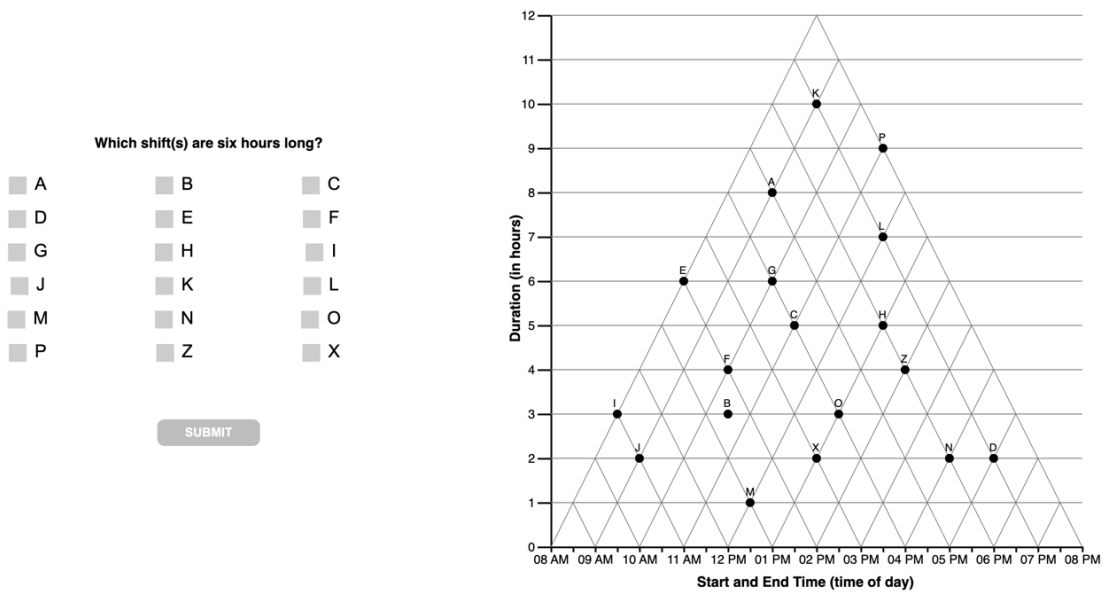


Figure 2.1: **Figure 1. Sample Graph Comprehension (Question # 6)**

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*.

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

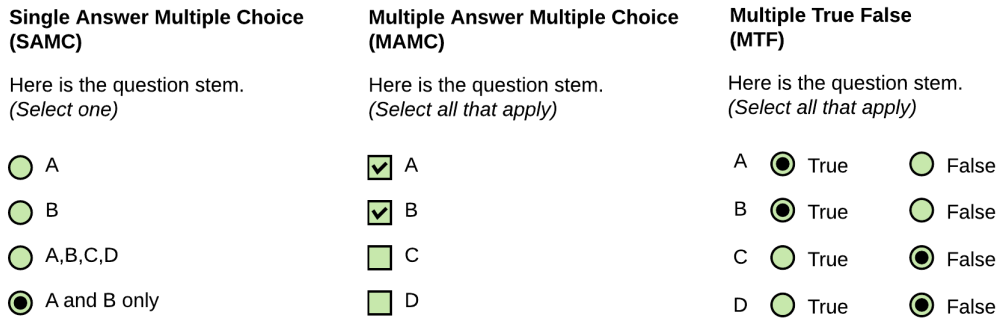


Figure 2.2: **Figure 2. SAMC (vs) MAMC (vs) MTF**

In this example (Figure **??**), 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/q, +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:

$$f = (p_s/p) - (q_s/q) \tag{2.14}$$
$$\tag{2.15}$$

```
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(NA), "") #handle empty response set gracefully by returning not
  ifelse( (p != 0), 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

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

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.

```
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",
              "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.

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

2.2 SCORE SGC DATA

In SGC_3A we are fundamentally interested in understanding how a participant interprets the presented graph (stimulus). The **graph comprehension task** 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:

TODO UPDATE SCORE DEFS

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.

2.2.1 Prepare Answer Keys

We start by importing three answer keys: (1) Q1 - Q5 [control condition], (2) Q1-Q5 [impasse condition], (3) Q6-15. Separate answer keys by condition are required for Q1-Q5 because the stimuli for each condition visualize a different underlying dataset (i.e. the graphs show datapoints in different positions). Q6-Q15 are identical across conditions. Each answer key includes a row for each question, and a column defining the subset of response options that correspond to different graph interpretations.

```
key_111_raw <- read_csv('data/keys/SGC3A_scaffold_111_key.csv') %>% mutate(condition = 111
key_121_raw <- read_csv('data/keys/SGC3A_scaffold_121_key.csv')%>% mutate(condition = 121,
cs = rep('c', 23) %>% str_c(collapse="") #create column spec
key_test_raw <- read_csv('data/keys/SGC3A_test_key.csv', col_types = cs)%>% mutate(conditi

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 (i.e. triangular, orthogonal, tversky) we must define these sets independently for each interpretation. For each question, the `keys_raw` dataframe gives us set N (all response options), and a set P (options that should be selected) for *each interpretation*. From these we must derive set Q for each interpretation.

- SET N, all response options (superset) . This set is the same across all interpretations (a property of the question) and is given in the answer key.
- SET P, $P \subset N$, the subset of options that **should** be selected (rewarded as +1/p) . This set differs by interpretation, and is given in the answer key.
- SET $A, A \subset N, A \sqcup P$, the subset of options that **should not** be selected, *but if they are, aren't penalized* (i.e. these options are ignored. Not rewarded, nor penalized). 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. This set differs by interpretation, and is given in the answer key (columns `REF_POINT` and `_also`).
- SET Q, the subset of options that **should not be selected** and are penalized (as -1/q). This set differs by interpretation and is not given in the answer key. We can derive set Q for each interpretation by $Q = N - (P \cup A)$

The next step in scoring is preparing interpretation-specific answer keys that specify sets N, P, A and Q for each question.

2.2.1.1 Triangular Key

First we construct a key set based on the 'Triangular' interpretation (i.e. the actually correct answers).

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

    #replace NAs
    TRI_allow = str_replace_na(TRI_allow,""),
    REF_POINT = str_replace_na(REF_POINT,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TRIANGULAR,
    set_p = str_replace_na(set_p,""),#replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TRI_allow,REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""),#replace na if empty
    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 = str_replace_na(s,"")
  # 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_tri[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 question, each option in N is included in either set P, A or Q (once and only once).

TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE

2.2.1.2 Orthogonal Key

Next we construct a key set based on the 'Orthogonal' interpretation.

```
verify_orth = c() #sanity check
##------------------------------------------------------------------------
##CONSTRUCT ORTHOGONAL KEY SET
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_orth <- keys_raw %>%
  select(Q, condition, OPTIONS, ORTHOGONAL, ORTH_allow, REF_POINT) %>%
  mutate(

    #replace NAs
    ORTH_allow = str_replace_na(ORTH_allow,""),
    REF_POINT = str_replace_na(REF_POINT,""),

    #P options that SHOULD be selected (rewarded)
    set_p = ORTHOGONAL,
    set_p = str_replace_na(set_p,""),#replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(ORTH_allow,REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""), #replace na if empty
    n_a = nchar(set_a),

    #N store all answer 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 = str_replace_na(s,"")
  # print(s)
  # 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()
  # print(tempunion)
  N = N %>% str_c(collapse="") %>% strsplit("") %>% unlist()

  verify_orth[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, cs)
```

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 question, each response in N is included in either set P, A or Q (once and only once).

TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE

2.2.1.3 Satisficing Key

Next we construct two keys based on the 'Satisficing' strategy. Satisficing involves selecting any data points within 0.5hr visual offset of the orthogonal interpretation of the graph (because no orthogonal response option is available). One key represents selecting a point slightly to the left of the orthogonal, and the other key represents selecting a point slightly to the right of the orthogonal.

```
verify_satisfice_right = c() #sanity check
##------------------------------------------------------------------------
##CONSTRUCT SATISFICE RIGHT KEY SET
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_satisfice_right <- keys_raw %>%
  select(Q, condition, OPTIONS, SATISFICE_right, REF_POINT) %>%
  mutate(
    #replace NAs
    REF_POINT = str_replace_na(REF_POINT,""),

    #P options that SHOULD be selected (rewarded)
    set_p = SATISFICE_right,
    set_p = str_replace_na(set_p,""), #replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""), #replace na if empty
    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_satisfice_right)) {

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

```
  P = keys_satisfice_right[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_satisfice_right[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 = str_replace_na(s,"")
  # print(s)
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected
  #save set to data frame
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_satisfice_right[x,'set_q'] = Q
  keys_satisfice_right[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_satisfice_right[x] = setequal(tempunion,N)
}

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


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



verify_satisfice_left = c() #sanity check
##------------------------------------------------------------------------
##CONSTRUCT SATISFICE left KEY SET
##------------------------------------------------------------------------
#1. DEFINE SETS N, P, A
keys_satisfice_left <- keys_raw %>%
  select(Q, condition, OPTIONS, SATISFICE_left, REF_POINT) %>%
  mutate(
```

```
    #replace NAs
    REF_POINT = str_replace_na(REF_POINT,""),

    #P options that SHOULD be selected (rewarded)
    set_p = SATISFICE_left,
    set_p = str_replace_na(set_p,""), #replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""), #replace na if empty
    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_satisfice_left)) {

  #UNWIND STRINGS FOR SETDIFF
  #n all answer options
  N = keys_satisfice_left[x,'set_n'] %>% pull(set_n) %>% strsplit("") %>% unlist()
  #p correct-select answer options
  P = keys_satisfice_left[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_satisfice_left[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 = str_replace_na(s,"")
  # print(s)
  # s = union(s,X) # + trapdoor
  Q = setdiff(N,s) # = penalized at -1/q when selected
  #save set to data frame
  Q = str_c(Q, collapse="")
  n_q = nchar(Q)
  keys_satisfice_left[x,'set_q'] = Q
```

```
  keys_satisfice_left[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_satisfice_left[x] = setequal(tempunion,N)
}

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


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

This leaves us a dataframe `keys_satisfice` 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 question, each response in N is included in either set P, A or Q (once and only once).

TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE

2.2.1.4 Tversky Keys

Next we construct the key set based on a partial-understanding strategy we refer to as 'Tversky'. We use the label Tversky as 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, arrows direct, boxes contain").

```
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, REF_POINT, TV_max, TV_max_allow) %>%
  mutate(
```

```
    #replace NAs
    REF_POINT = str_replace_na(REF_POINT,""),
    TV_max = str_replace_na(TV_max,""),
    TV_max_allow = str_replace_na(TV_max_allow,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TV_max,
    set_p = str_replace_na(set_p,""), #replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TV_max_allow,REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""), #replace na if empty
    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 = str_replace_na(s,"")
  # 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, REF_POINT, TV_start, TV_start_allow) %>%
  mutate(

    #replace NAs
    REF_POINT = str_replace_na(REF_POINT,""),
    TV_start = str_replace_na(TV_start,""),
    TV_start_allow = str_replace_na(TV_start_allow,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TV_start,
    set_p = str_replace_na(set_p,""), #replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TV_start_allow,REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""), #replace na if empty
    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 = str_replace_na(s,"")
  # 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
```

```r
keys_tversky_end <- keys_raw %>%
  select(Q, condition, OPTIONS, REF_POINT, TV_end, TV_end_allow) %>%
  mutate(

    #replace NAs
    REF_POINT = str_replace_na(REF_POINT,""),
    TV_end = str_replace_na(TV_end,""),
    TV_end_allow = str_replace_na(TV_end_allow,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TV_end,
    set_p = str_replace_na(set_p,""), #replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TV_end_allow,REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""), #replace na if empty
    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 = str_replace_na(s,"")
  # 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, REF_POINT, TV_dur, TV_dur_allow) %>%
  mutate(

    #replace NAs
    REF_POINT = str_replace_na(REF_POINT,""),
    TV_dur = str_replace_na(TV_dur,""),
    TV_dur_allow = str_replace_na(TV_dur_allow,""),

    #P options that SHOULD be selected (rewarded)
    set_p = TV_dur,
    set_p = str_replace_na(set_p,""), #replace na if empty
    n_p = nchar(set_p), #number of true-select options

    #A options that are ignored if selected
    set_a = str_c(TV_dur_allow,REF_POINT, sep=""),
    set_a = str_replace_na(set_a,""), #replace na if empty
    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 = str_replace_na(s,"")
  # 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 question, each response in N is included in either set P, A or Q (once and only once).

TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE

TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE

TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE

TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE

```
#cleanup
rm(verify_tri, verify_orth, verify_max, verify_tversky_duration, verify_tversky_end, verif
```

2.2.2 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 (e.g. `keys_orth`, `keys_tri`, etc…), 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 in `df_items`, and can be used to determine which graph interpretation the subject held.

Specifically, the following scores are calculated for each item:

Interpretation Subscores

- `score_TRI` How consistent is the response with the **triangular**interpretation?
- `score_ORTH` How consistent is the response with the **orthogonal**interpretation?

- `score_SATISFICE` is calculated by taking the maximum value of :
  - `score_SAT_left` How consistent is the response with the **(left side) Satisficing** interpretation?
  - `score_SAT_right` How consistent is the response with the **(right side) Satisficing** interpretation
- `score_TVERSKY` is calculated by taking the maximum value of:
  - `score_TV_max` How consistent is the response with the **(maximal) Tversky** interpretation?
  - `score_TV_start` How consistent is the response with the **(start-time) Tversky** interpretation?
  - `score_TV_end` How consistent is the response with the **(end-time) Tversky**i nterpretation?
  - `score_TV_duration` How consistent is the response with the **(duration) Tversky** interpretation?
- `score_REF` Did the response select only the **reference point**?
- `score_BOTH` How consistent is the response with **both** the orthogonal and triangular interpretations?

Absolute Scores

- `score_ABS` Is the response strictly correct? (triangular interpretation)
- `score_niceABS` Is the response strictly correct? (triangular interpretation, not penalizing ref points)

```
backup <- read_rds('data/1-study-level/sgc3a_items.rds')
df_items <- read_rds('data/1-study-level/sgc3a_items.rds')
```

```
calc_sub_score2 <- function(question, cond, response, keyframe){

  # print(paste(question, cond, response))

    #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 == cond) %>% select(set_p
    q =  keyframe %>% filter(Q == question) %>% filter(condition == cond) %>% select(set_q
    pn = keyframe %>% filter(Q == question) %>% filter(condition == cond) %>% select(n_p)
```

```
    qn = keyframe %>% filter(Q == question) %>% filter(condition == cond) %>% select(n_q)

    # print(p)
    # print(q)
    # print(paste("pn ",pn))
    # print(paste("qn ",qn))

  } 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

  #if response is not empty, split apart response for set comparison
    if(response != "")
    { response = response %>% str_split("") %>% unlist()}

  #set comparisons
  ps = length(intersect(response,p))
  # print(paste("correct selected" ,ps))
  qs = length(intersect(response,q))
  # print(paste("incorrect selected", qs))
  # 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) %>% unlist() %>% as.numeric()
  # print(x)
  #cleanup
  rm(p,q,pn,qn,ps,qs)

  return(x)
}

#CALCULATE THE REFERENCE SCORES
calc_ref_score2 <- function(question, cond, response){
```

```
  #1. GET reference point from REF_POINT column in raw keys
  if (question < 6) {
    ref_p = keys_raw %>% filter(Q == question) %>% filter(condition == cond) %>% select(RE
  } else {
    ref_p = keys_raw %>% filter(Q == question) %>%select(REF_POINT) %>%
      pull(REF_POINT) %>% str_split("") %>% unlist()
  }

    #2. Is the response PRECISELY the REFERENCE POINT?
    x = identical(ref_p,response)
    x = as.numeric(x)

    # paste("ref: ",ref_p)
    # paste("response: ",response)
    # paste("x: ",ref_p == response)

    #cleanup
    rm(ref_p, response, question, cond)
    return(x) #1 = match, 0 = not match
}

#CALCULATE SCORE BASED ON UNION OF ORTH & TRI (SUBJECT SELECTS BOTH ANSWERS )
calc_both_score2 <- function(question, cond, response){


  #TRAPDOOR
  #since no orth responses exist for impasse condition q1 - q5, set to 0
  if (question < 6 & cond == 121) {x = NA}

  #ELSE
  #calculate union of ORTH and TRI
  else {
    if (question < 6 & cond == 111) #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 == cond) %>% select
     orth_p = keys_orth %>% filter(Q == question) %>% filter(condition == cond) %>% select
     set_n =  keys_tri %>%  filter(Q == question) %>% filter(condition == cond) %>% select
     #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

  #if response is not empty, split apart response for set comparison
    if(response != "")
    { response = response %>% str_split("") %>% unlist()}

    both_ps = length(intersect(response,both_p))
    both_qs = length(intersect(response,both_q))


  #STEP 3 CALC f_partialP schema SCORE FOR THIS INTERSECTION
  x = f_partialP(both_ps,both_pn,both_qs,both_qn)%>% unlist() %>% as.numeric()

  #cleanup
  rm(both_p,both_q,both_pn,both_qn,both_ps,both_qs, question, cond, response )
  }

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


#REORDER THE CHARACTERS IN A STRING
reorder_inplace <- function(x)
{
```

```
  y =  x %>% str_split("") %>% unlist() %>% sort() %>% str_c(collapse="")
  return (y)
}
```

note: this cell takes approximately 30 minutes to run on the full df_items dataframe with 4950 records

```
#RUN THIS <OR> THE CALCULATE-SCORES-FORLOOP [not both]

#ALPHEBETIZE RESPONSE
df_items$response = pbmapply(reorder_inplace, df_items$response)

# #STRATEGY SUBSCORES
df_items$score_TRI = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_items$re
df_items$score_ORTH = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_items$r

df_items$score_SAT_left = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_ite
df_items$score_SAT_right = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_it

df_items$score_TV_max = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_items
df_items$score_TV_start = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_ite
df_items$score_TV_end = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_items
df_items$score_TV_duration = pbmapply(calc_sub_score2, df_items$q, df_items$condition, df_

#TODO SPECIAL SUBSCORES
df_items$score_REF = pbmapply(calc_ref_score2, df_items$q, df_items$condition, df_items$re
df_items$score_BOTH = pbmapply(calc_both_score2, df_items$q, df_items$condition, df_items$


#ABSOLUTE SCORES
df_items$score_ABS = as.integer(df_items$correct)
df_items$score_niceABS  <- as.integer((df_items$score_TRI == 1))

# alt_scored = df_items
```

TODO: generate heat maps of Q9. Same answer but very different optimal operation paths!

2.2.3 Derive Interpretation

Finally, we compare the interpretation subscores and decide which is highest. This indicates the interpretation that the individual's response most closely approximates. This algorithm

makes assigns one of the following values to each item response:

- Orthogonal
- Triangular
- Tversky
- both Tri + Orth
- reference
- blank
- frenzy
- ? TODO ADJUST 'both' to select for both tri/satisfice or both tri/orth? See Q1 121 'CO'

```
threshold_range = 0.5 #set required variance in subscores to be discriminant
threshold_frenzy = 4

for (x in 1:nrow(df_items)) {

  #CALCULATE MAX TVERSKY SUBSCORE
  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)
  t.long[t.long == ""] = NA #replace empty scores with NA so we can ignore them
  if(length(unique(t.long$value)) == 1 ){
    if(is.na(unique(t.long$value))){
      df_items[x,'score_TVERSKY'] = NA
      df_items[x,'tv_type'] = NA
    }
  } else {
    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']
  }

  #CALCULATE MAX SATISFICING SUBSCORE
  t = df_items[x,] %>% select(score_SAT_left, score_SAT_right)
  t.long = gather(t,score, value, 1:2)
  t.long[t.long == ""] = NA #replace empty scores
  if(length(unique(t.long$value)) == 1 ){
    if(is.na(unique(t.long$value))){
      df_items[x,'score_SATISFICE'] = NA
      df_items[x,'sat_type'] = NA
```

```
  }
} else {
  df_items[x,'score_SATISFICE'] = as.numeric(max(t.long$value,na.rm = TRUE))
  df_items[x,'sat_type'] = t.long[which.max(t.long$value),'score']
}

#NOW CALCULATE RANGE AMONG SUBSCORES
#order of this selection matters in breaking ties!
t = df_items[x,] %>% select(score_TRI, score_TVERSKY, score_SATISFICE, score_ORTH)
t.long = gather(t,score, value, 1:4)
t.long[t.long == ""] = NA

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

#calculate the range between highest and lowest scores
r = as.numeric(range(t.long$value,na.rm = TRUE))
r = diff(r)
df_items[x,'range'] = r

#DISCRIMINANT BETWEEN SUBSCORES TO PREDICT BEST FIT INTERPRETATION

if (r < threshold_range) {
    #then we can't predict the interpretation, leave it as "?"
  df_items[x,'best'] = "?"
} else {
    p =  df_items[x,'top_type']
    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"
    } else if(p == "score_SATISFICE") {df_items[x,'best'] = "Satisfice"}
}

#CHECK SPECIAL SITUATIONS

#BOTH TRI AND ORTH?
if (!is.na(df_items[x,'score_BOTH'])) { #only check if both is not null
    if( df_items[x,'score_BOTH'] == 1) {
      df_items[x,'best'] = "both tri + orth"}
}
```

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

  #IS FRENZY?
  if( df_items[x,'num_o'] > threshold_frenzy) {
      df_items[x,'best'] = "frenzy"
  }

  #IS REF POINT?
  if (!is.na(df_items[x,'score_REF'])) { #only check if the score is NOT null
      if( df_items[x,'score_REF'] == 1) {
          df_items[x,'best'] = "reference"
      }
  }

}#end loop

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

#set order of levels
df_items$interpretation <- factor(df_items$best,
                                  levels = c("Triangular", "Tversky",
                                             "Satisfice", "Orthogonal", "reference", "both


df_items$sorted_interpretation <- factor(df_items$best,
                                  levels = c("Triangular", "Tversky", "both tri + orth",
                                             "reference","frenzy","blank","?",
                                             "Satisfice", "Orthogonal"))

#recode as numeric inase they are char
# df_items$score_TV_duration <- df_items$score_TV_duration %>% as.numeric()
# df_items$score_SATISFICE <- df_items$score_SATISFICE %>% as.numeric()
```

2.2.4 Derive Discriminant Score

"Orthogonal" = -1, "Satisfice" = -1, "Triangular" = 1, "Tversky" = 0.5, "both tri + orth" = 0.5, "reference" = 0, "blank" = 0, "frenzy" = 0, "?" = 0)

TODO should 'both' be coded as 0 or 1 or 0.5?

TODO write description TODO recode with satisfice as -1

```
df_items$score_SCALED <- recode(df_items$interpretation,
                            "Orthogonal" = -1,
                            "Satisfice" = -1,
                            "Triangular" = 1,
                            "Tversky" = 0.5,
                            "both tri + orth" = 0.5,
                            "reference" = 0,
                            "blank" = 0,
                            "frenzy" = 0,
                            "?" = 0)
```

2.3 EXPLORE RESPONSES

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

2.3.1 Scaffold Phase

The first five questions constitute the 'scaffold' (or learning) phase, where participants see a different version of the stimulus (specifically a different dataset is visualized) invoking a different experimental condition.

2.3.1.1 Question #1

2.3.1.1.1 Q1. Control Condition

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).

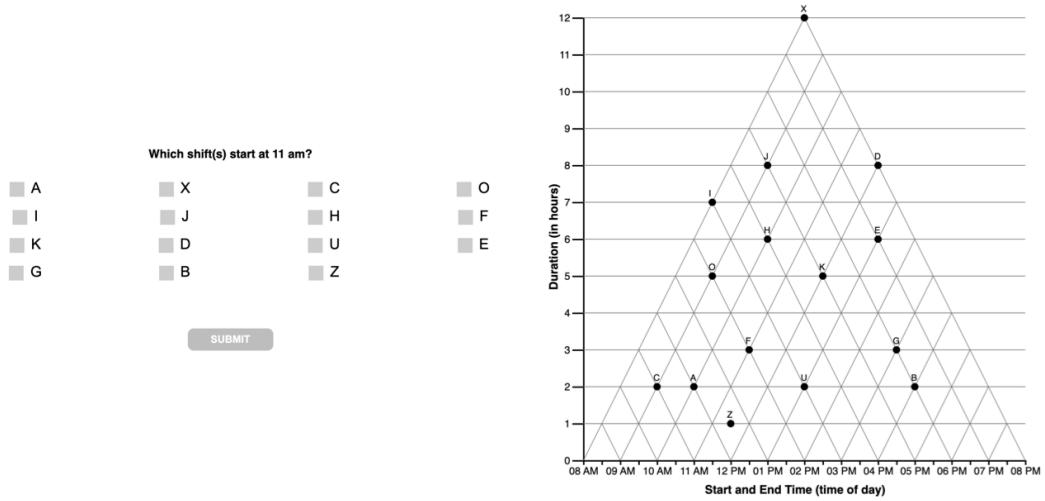


Figure 2.3: Question 1 — Control Condition

```
q <- keys_raw %>% filter(condition == 111) %>% filter(Q==1)
ignore <- q %>% select("REF_POINT")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL", "SATISFICE_left", "SATISFICE_right","T
ves <- q %>% mutate(
  SATISFICE_left_allow = "",
  SATISFICE_right_allow = ""
) %>% select("TRI_allow", "ORTH_allow", "SATISFICE_left_allow","SATISFICE_right_allow", "T
options <- q %>% select("OPTIONS")
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Satisficing [left]", "Satisficing [right]", "Tvers
            "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
```

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 **??**). `n` in-

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
Satisficing [left]		
Satisficing [right]		
Tversky [maximal]	CF	Z
Tversky [start diagonal]	F	Z
Tversky [end diagonal]	C	
Tversky [duration line]		

Note: 15 response options: AIKGXJDBCHUZOFE

dicates the number of participants who gave this response, while `interpretation` indicates the *graph interpretation* most consistent with that response. At the right of this table are the Absolute, followed by Partial Credit subscores for each response. NA indicates that there is no score calculated (occurs when there is no subset of response options that accord with that interpretation for this question).

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.

```
title <- "Frequency of Selected Response Options for Question #1 (Control Condition)"
names = c("response","n","interpretation","absolute","tri","tversky","satisfice","orthogon

df_items %>% filter(q == 1 & condition == 111) %>% group_by(response) %>%
  dplyr::summarise( count = n(),
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
                    orthogonal =  unique(score_ORTH),
                    satisficing =  unique(score_SATISFICE),
                    tversky = unique(score_TVERSKY),
                    interpretation = unique(interpretation),
                    scaled = unique(score_SCALED)) %>%
  arrange(interpretation, desc(count)) %>%
  select(response, count, interpretation, nice,
         triangular, tversky, satisficing, orthogonal, scaled) %>%
  kbl(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Score" = 1, "Interpretation Scores"=4, "Discriminant
  pack_rows("Triangular", 1, 1) %>%
  pack_rows("Lines-Connect", 2, 2) %>%
```

```
    pack_rows("Orthogonal", 3, 3) %>%
    pack_rows("Other", 4, 4)  %>%
    pack_rows("Unknown", 5, 7)  %>%
    footnote(general = "n = number of responses in sample",
             general_title = "Note: ",footnote_as_chunk = T)
```

\begin{table}

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

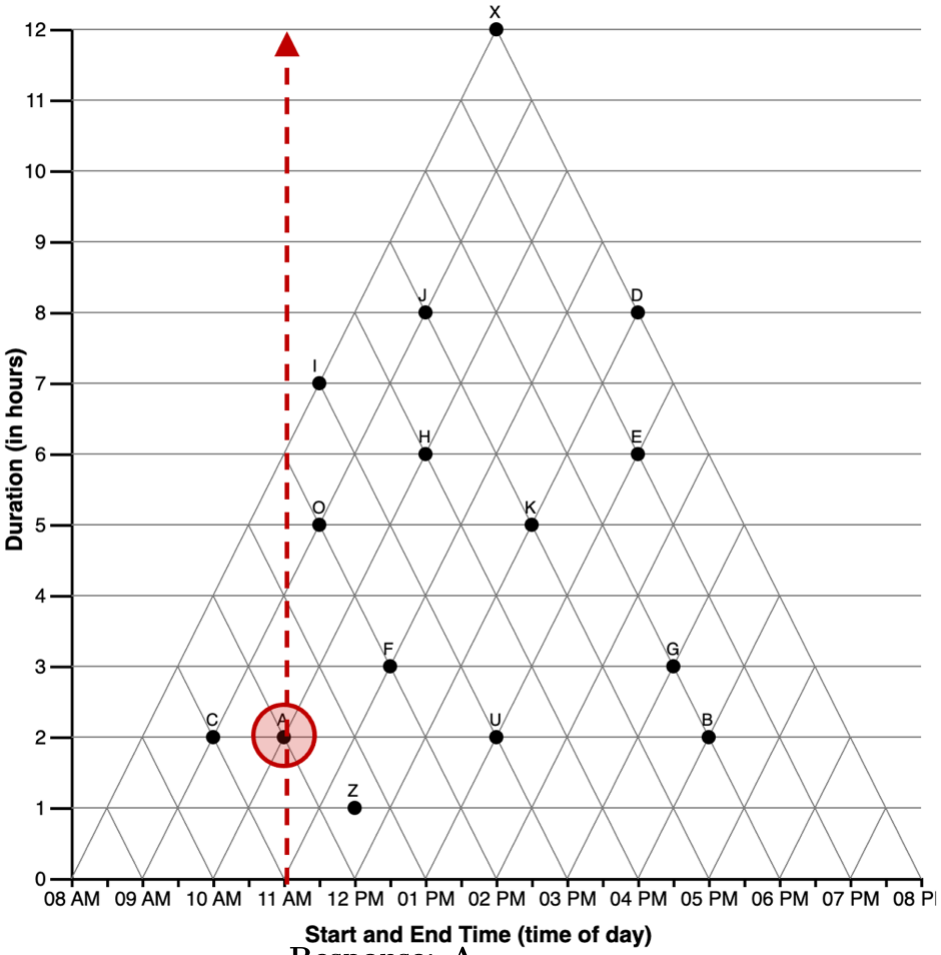
			Strict Score	Interpretation Scores				Discriminant
response	n	interpretation	absolute	tri	tversky	satisfice	orthogonal	scaled score
Triangular								
F	22	Triangular	1	1.000	1.000	NA	-0.083	1.0
Lines-Connect								
CF	3	Tversky	0	0.923	1.000	NA	-0.167	0.5
Orthogonal								
A	129	Orthogonal	0	-0.077	-0.071	NA	1.000	-1.0
Other								
AF	1	both tri + orth	0	0.923	0.923	NA	0.917	0.5
Unknown								
DIJ	1	?	0	-0.231	-0.214	NA	-0.167	0.0
X	1	?	0	-0.077	-0.071	NA	-0.083	0.0
Z	1	?	0	0.000	0.000	NA	-0.083	0.0

Note: n = number of responses in sample

\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 `?` .

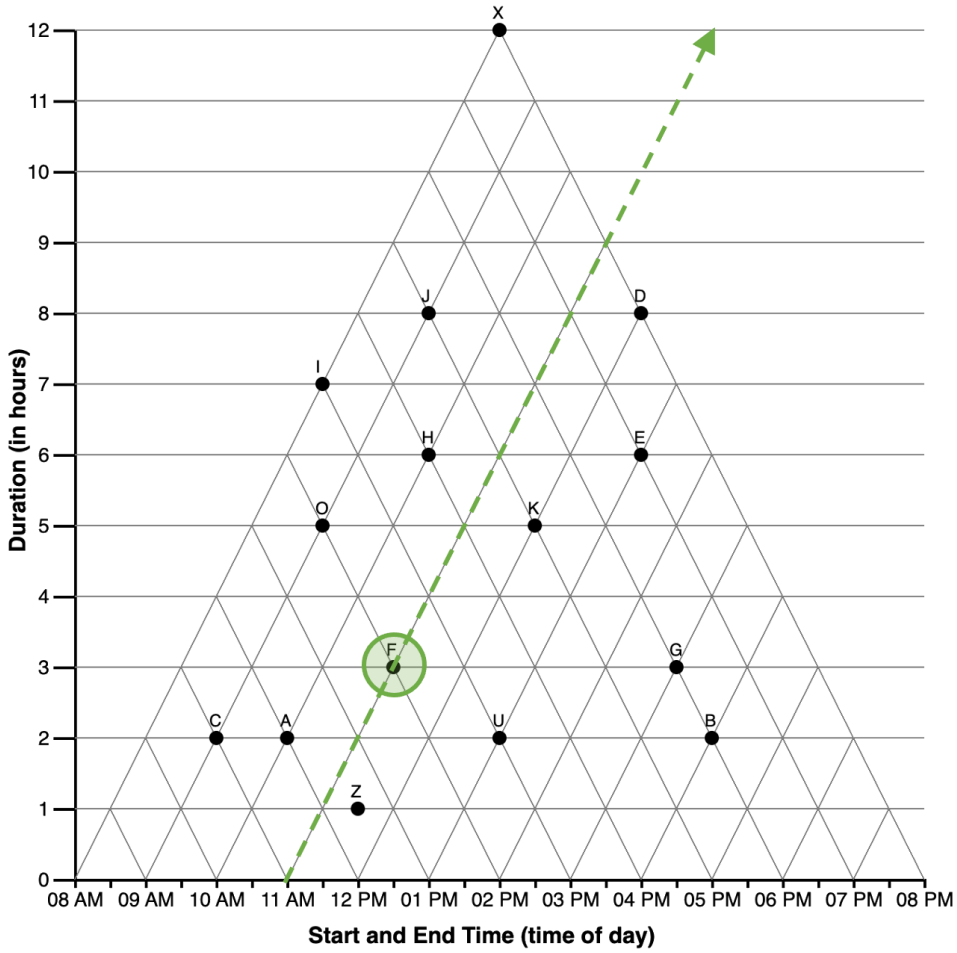
Which shifts start at
11am?



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**.

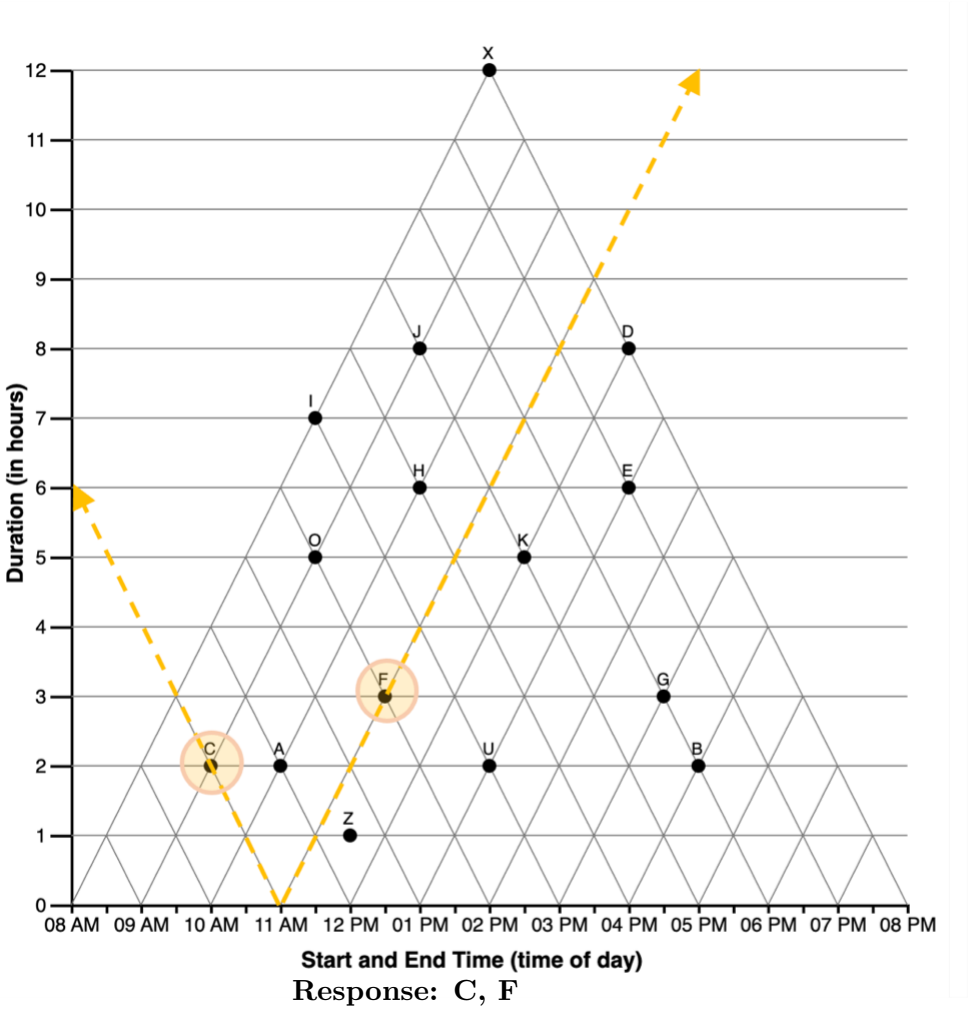
Which shifts start at
11am?



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**.

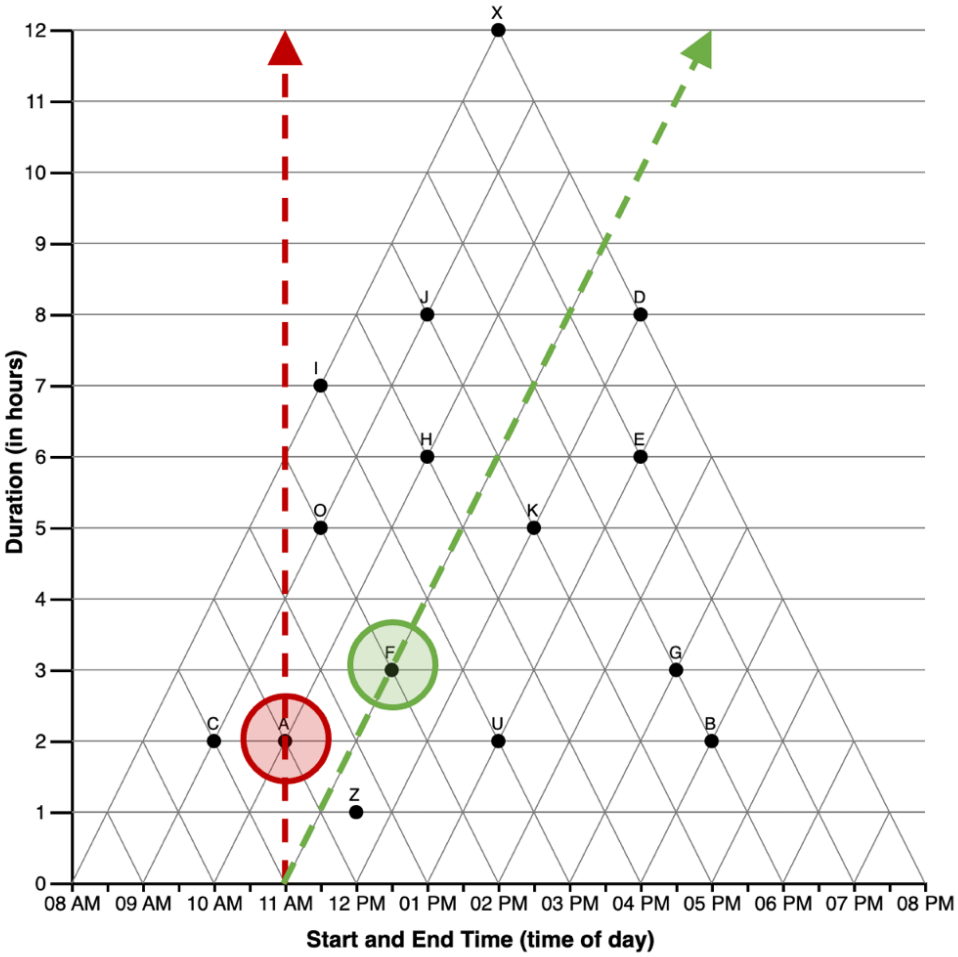
Which shifts start at
11am?



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.

Which shifts start at
11am?

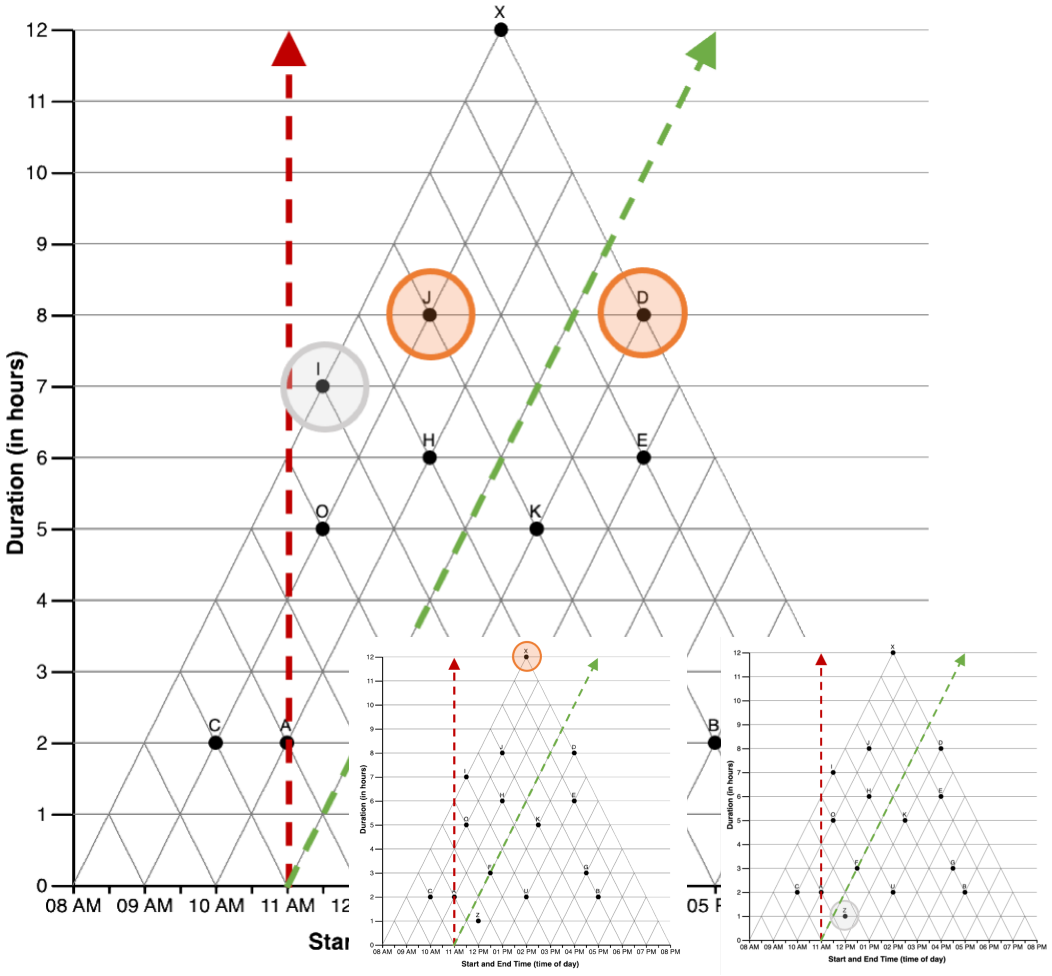


Response: A , F

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

Three responses were given that were not consistent with any of the identified interpretations. *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.*

D I J	**X**	**Z** TODO find this person, did they subsequently give tri answers?



2.3.1.1.2 Q1. Impasse Condition

Next we explore 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 == 121) %>% filter(Q==1)
ignore <- q %>% select("REF_POINT")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL", "SATISFICE_left", "SATISFICE_right","T
```

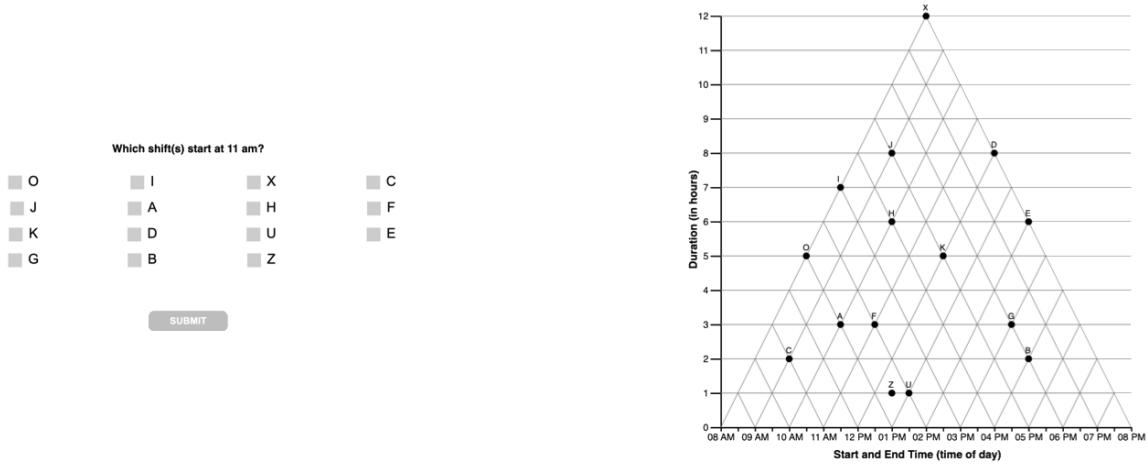


Figure 2.4: Question 1 — Impasse Condition

```
ves <- q %>% mutate(
  SATISFICE_left_allow = "",
  SATISFICE_right_allow = ""
) %>% select("TRI_allow", "ORTH_allow", "SATISFICE_left_allow","SATISFICE_right_allow", "T
options <- q %>% select("OPTIONS")
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Satisficing [left]", "Satisficing [right]", "Tvers
            "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 Impasse 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 there **is no orthogonal answer** for this question. This is the purpose of the impasse condition, to remove the possibility of selecting the orthogonal answer, we expect learners will be more likely to restructure their understanding of the coordinate system, and arrive at a correct (triangular) interpretation.

Table 2.7: Answer Key | Q1 Impasse Condition : Which shift(s) start at 11 am?

interpretation	answer	not penalized
Triangular	F	
Orthgonal		
Satisficing [left]	O	
Satisficing [right]	AI	
Tversky [maximal]	CF	
Tversky [start diagonal]	F	
Tversky [end diagonal]	C	
Tversky [duration line]		

Note: 15 response options: AIKGXJDBCHUZOFE

```
title <- "Frequency of Selected Response Options for Question #1 (Impasse Condition)"
names = c("response","n","interpretation","absolute","tri","tversky","satisfice","orthogon

df_items %>% filter(q == 1 & condition == 121) %>% group_by(response) %>%
  dplyr::summarise( count = n(),
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
                    orthogonal =  unique(score_ORTH),
                    satisficing =  unique(score_SATISFICE),
                    tversky = unique(score_TVERSKY),
                    interpretation = unique(interpretation),
                    scaled = unique(score_SCALED)) %>%
  arrange(interpretation, desc(count)) %>%
  select(response, count, interpretation, nice,
         triangular, tversky, satisficing, orthogonal, scaled) %>%
  kbl(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Score" = 1, "Interpretation Scores"=4, "Discriminant
  pack_rows("Triangular", 1, 1) %>%
  pack_rows("Lines-Connect", 2, 4) %>%
  pack_rows("Satisfice", 5, 9) %>%
  pack_rows("Other", 10, 10) %>%
  pack_rows("Unknown", 11, 12) %>%
  footnote(general = "n = number of responses in sample",
           general_title = "Note: ",footnote_as_chunk = T)
```

\begin{table}

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

			Strict Score	Interpretation Scores				Discriminant
response	n	interpretation	absolute	tri	tversky	satisfice	orthogonal	scaled score
Triangular								
F	49	Triangular	1	1.000	1.000	-0.071	NA	1.0
Lines-Connect								
CF	14	Tversky	0	0.929	1.000	-0.143	NA	0.5
C	3	Tversky	0	-0.071	1.000	-0.071	NA	0.5
CO	1	Tversky	0	-0.143	0.929	0.929	NA	0.5
Satisfice								
O	28	Satisfice	0	-0.071	-0.071	1.000	NA	-1.0
AI	9	Satisfice	0	-0.143	-0.143	1.000	NA	-1.0
A	4	Satisfice	0	-0.071	-0.071	0.500	NA	-1.0
AO	2	Satisfice	0	-0.143	-0.143	0.929	NA	-1.0
I	2	Satisfice	0	-0.071	-0.071	0.500	NA	-1.0
Other								
	57	blank	0	0.000	0.000	NA	NA	0.0
Unknown								
E	2	?	0	-0.071	-0.071	-0.071	NA	0.0
X	1	?	0	-0.071	-0.071	-0.071	NA	0.0

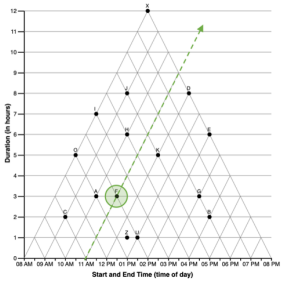
Note: n = number of responses in sample

\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 `?` .

TODO ADJUST 'both' to select for both tri/satisfice or both tri/orth

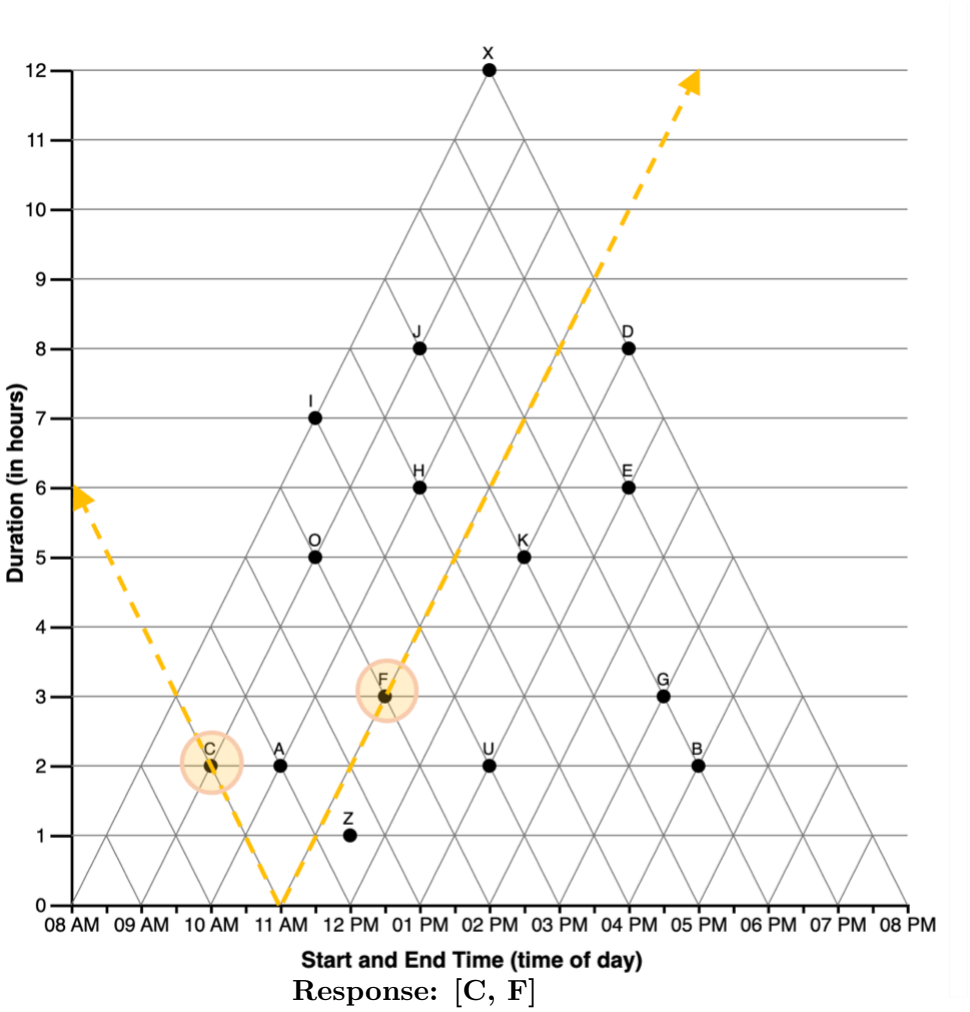
Which shifts start at 11am?



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**.

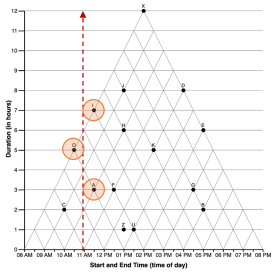
Which shifts start at
11am?



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?

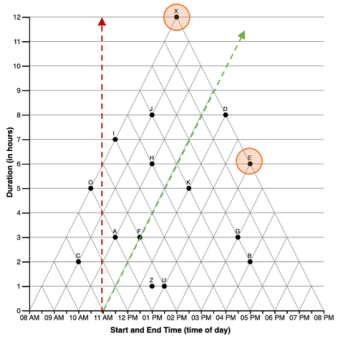


Responses: [AOI]

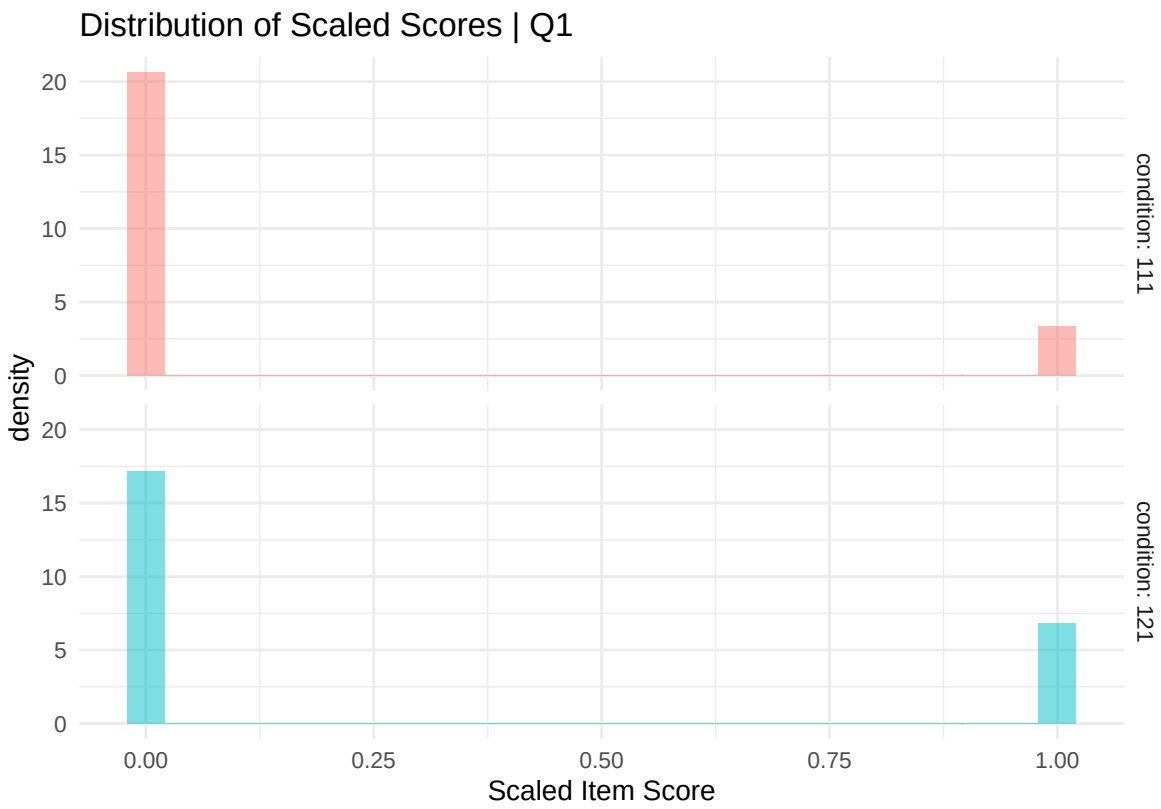
- indicates a **satisficing** strategy
- Consistent with the reader identifying the datapoints nearest to the orthogonal projection from the reference point point

Two responses were given that were not consistent with any of the identified interpretations.

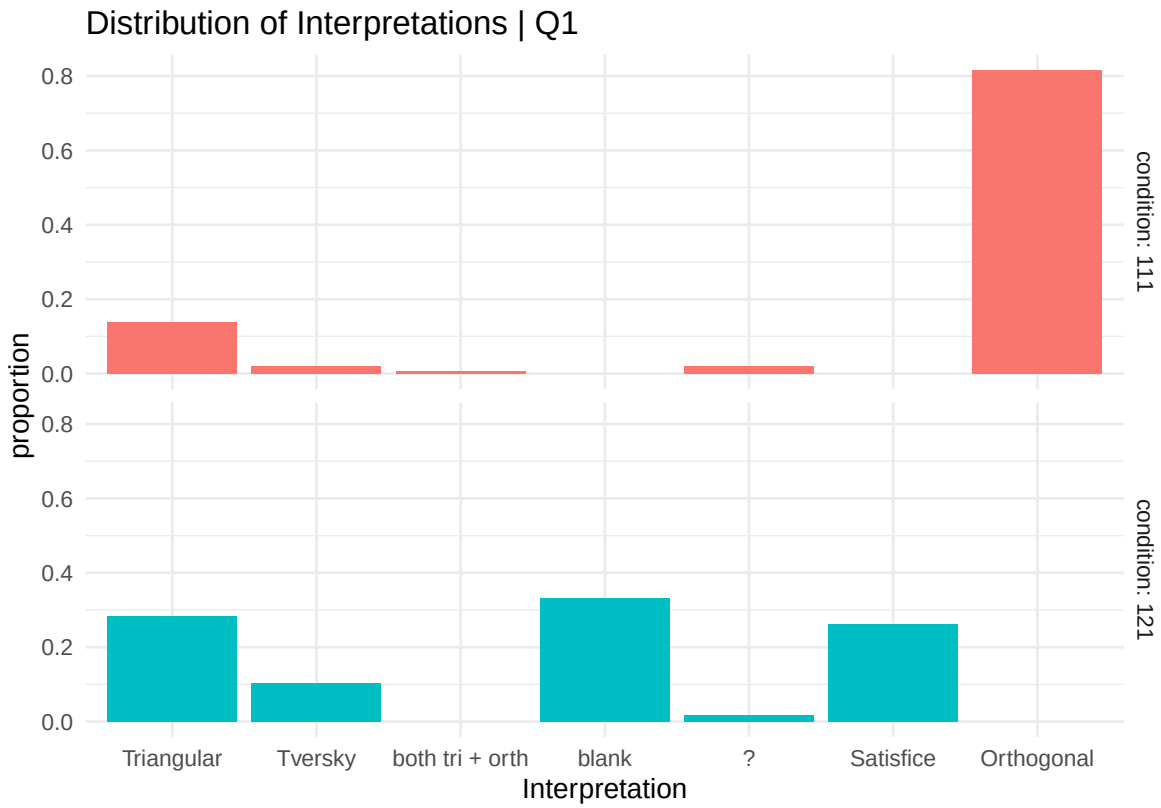
[E],[X]




```
gf_dhistogram(~ score_niceABS, fill = ~condition, data = df_items %>% filter(q ==1)) %>%
  gf_facet_grid( condition ~ ., labeller = label_both) +
  labs( x = "Scaled Item Score", title = "Distribution of Scaled Scores | Q1 ") +
  theme_minimal() + theme(legend.position = "blank")
```




```
gf_props(~sorted_interpretation, fill = ~condition, data = df_items %>% filter(q ==1)) %>%
  gf_facet_grid( condition ~ ., labeller = label_both) +
  labs( x = "Interpretation", title = "Distribution of Interpretations | Q1 ") +
  theme_minimal() + theme(legend.position = "blank")
```



2.3.1.2 Question #2

2.3.1.2.1 Q2. Control Condition

```
q <- keys_raw %>% filter(condition == 111) %>% filter(Q==2)
ignore <- q %>% select("REF_POINT")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL", "SATISFICE_left", "SATISFICE_right","T
ves <- q %>% mutate(
  SATISFICE_left_allow = "",
  SATISFICE_right_allow = ""
) %>% select("TRI_allow", "ORTH_allow", "SATISFICE_left_allow","SATISFICE_right_allow", "T
options <- q %>% select("OPTIONS")
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Satisficing [left]", "Satisficing [right]", "Tvers
            "Tversky [end diagonal]", "Tversky [duration line]")
d = tibble(interpretation = scores, answer = answers, allowed=ves)
d$answer <- replace_na(d$answer, "")
```

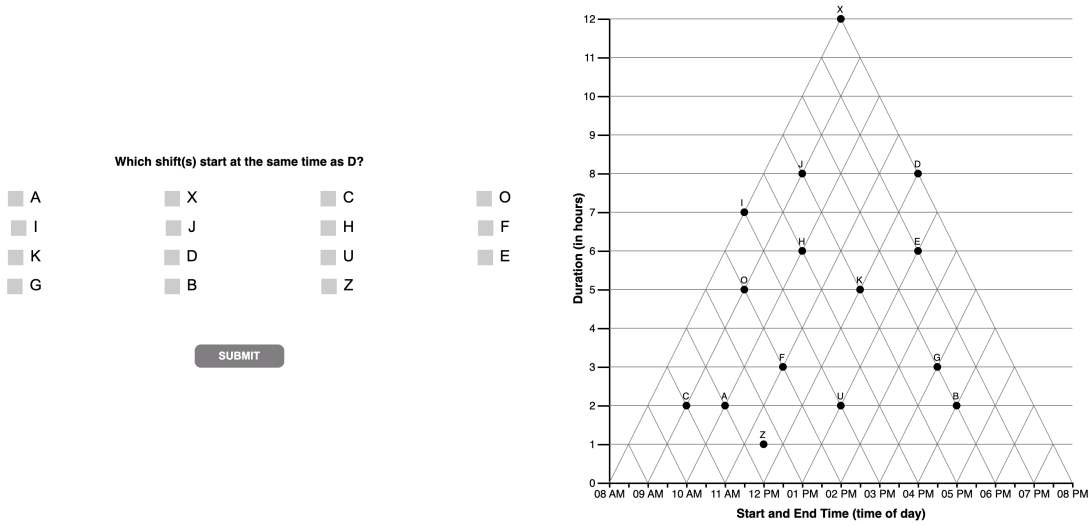


Figure 2.5: Q2—Control Condition

```
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
```

```
title <- "Frequency of Selected Response Options for Question #2 (Control Condition)"
names = c("response","n","interpretation","absolute","tri","tversky","satisfice","orthogon

df_items %>% filter(q == 2 & condition == 111) %>% group_by(response) %>%
  dplyr::summarise( count = n(),
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
                    orthogonal =  unique(score_ORTH),
                    satisficing =  unique(score_SATISFICE),
                    tversky = unique(score_TVERSKY),
                    interpretation = unique(interpretation),
                    scaled = unique(score_SCALED)) %>%
  arrange(interpretation, desc(count)) %>%
```

Table 2.10: 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
Satisficing [left]		
Satisficing [right]		
Tversky [maximal]	AKJX	Z
Tversky [start diagonal]	AK	Z
Tversky [end diagonal]	X	
Tversky [duration line]	J	

Note: 15 response options: AIKGXJDBCHUZOFE

```
  select(response, count, interpretation, nice,
         triangular, tversky, satisficing, orthogonal, scaled) %>%
  kbl(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Score" = 1, "Interpretation Scores"=4, "Discriminant
  pack_rows("Triangular", 1, 2) %>%
  pack_rows("Lines-Connect", 3, 4) %>%
  pack_rows("Orthogonal", 5, 7) %>%
  pack_rows("Other", 8, 8)  %>%
  pack_rows("Unknown", 9, 10)  %>%
  footnote(general = "n = number of responses in sample",
           general_title = "Note: ",footnote_as_chunk = T)
```

\begin{table}

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

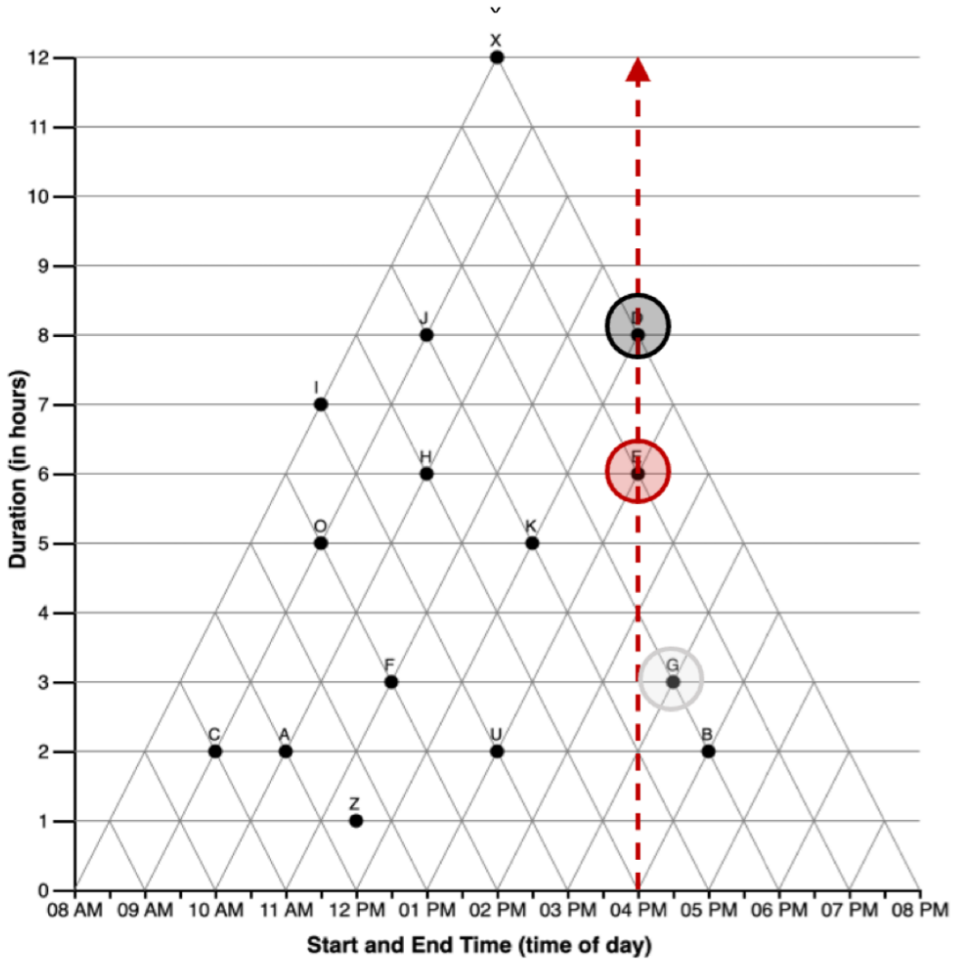
			Strict Score	Interpretation Scores				Discriminant
response	n	interpretation	absolute	tri	tversky	satisfice	orthogonal	scaled score
Triangular								
K	24	Triangular	1	1.000	0.500	NA	-0.083	1.0
DK	1	Triangular	1	1.000	0.500	NA	-0.083	1.0
Lines-Connect								
J	4	Tversky	0	-0.083	1.000	NA	-0.083	0.5
AK	1	Tversky	0	0.917	1.000	NA	-0.167	0.5
Orthogonal								
E	121	Orthogonal	0	-0.083	-0.077	NA	1.000	-1.0
DE	3	Orthogonal	0	-0.083	-0.077	NA	1.000	-1.0
EG	1	Orthogonal	0	-0.167	-0.154	NA	1.000	-1.0
Other								
D	1	reference	0	0.000	NA	NA	0.000	0.0
Unknown								
B	1	?	0	-0.083	-0.077	NA	-0.083	0.0
C	1	?	0	-0.083	-0.077	NA	-0.083	0.0

Note: n = number of responses in sample

\end{table}

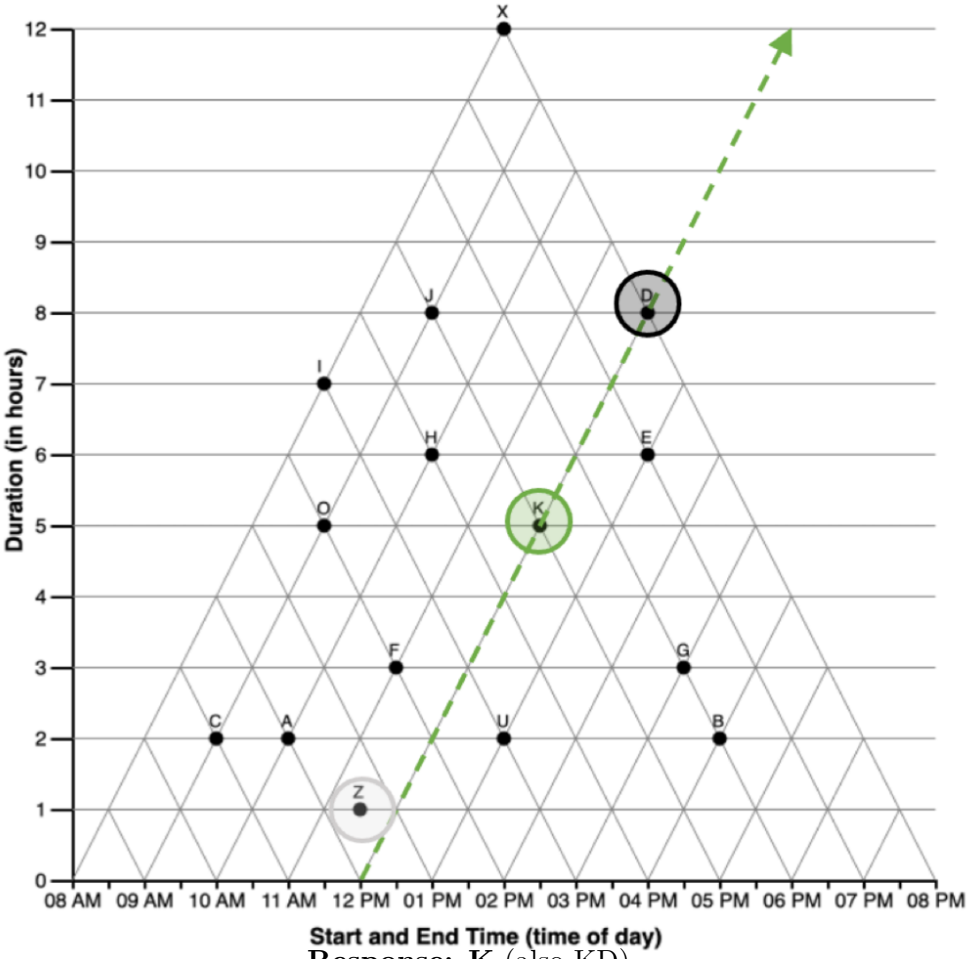
Again, we see that most subjects selected a response consistent with one of the identified interpretations. (note, when the question stem includes a data point rather than time as reference, we do not penalize respondents for selecting the reference data point *in addition* to an interpretation consistent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



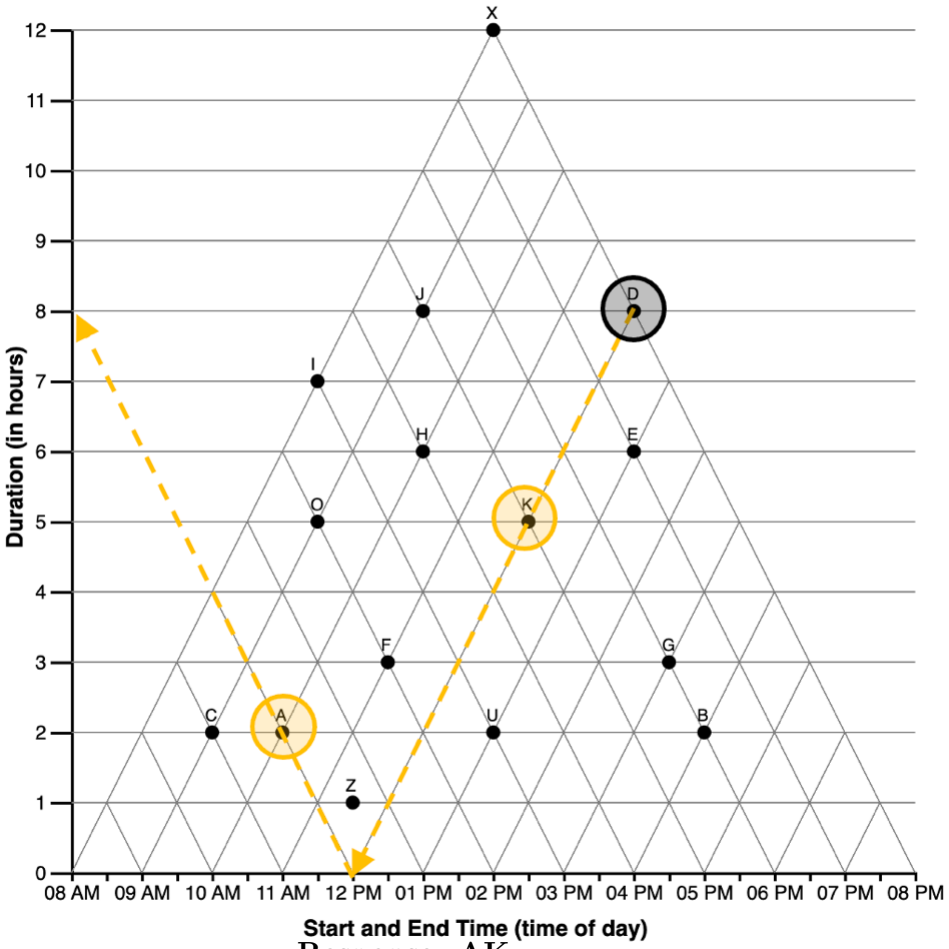
Reponse: **E** (also EG, DE)

- 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**.



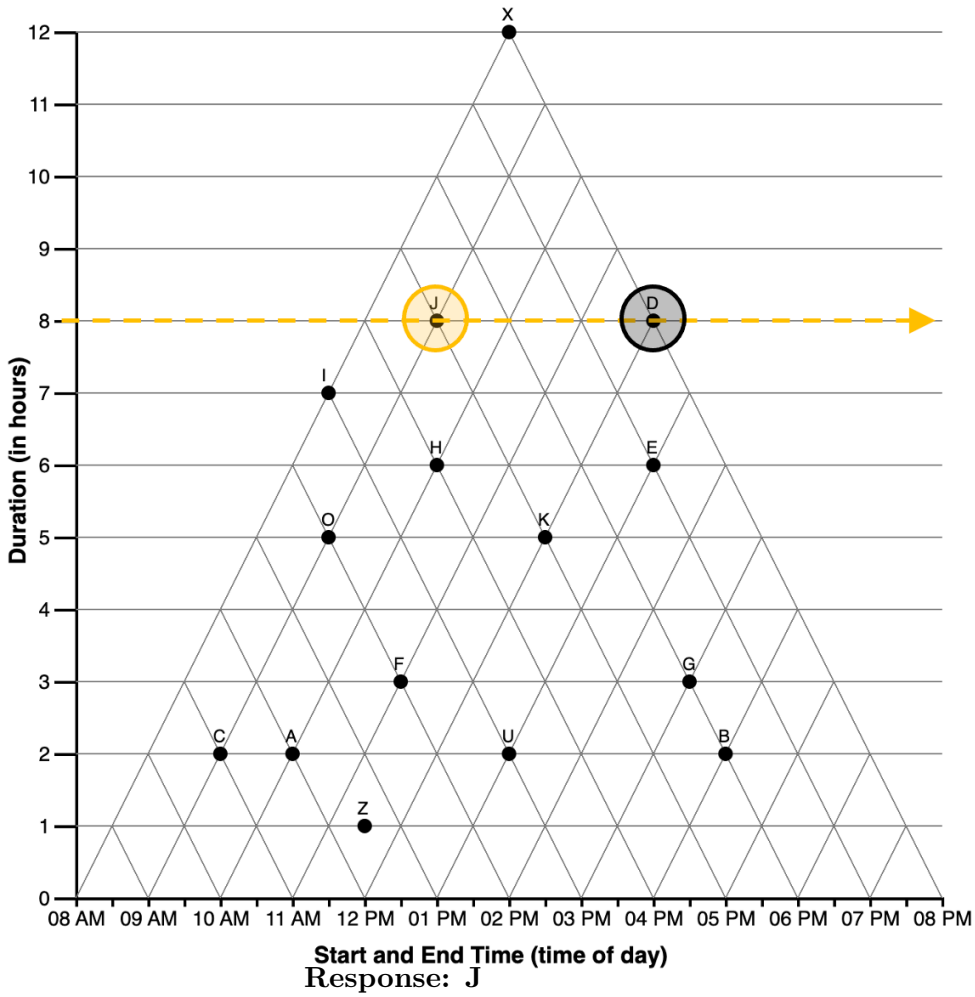
Response: **K** (also KD)

- 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**.



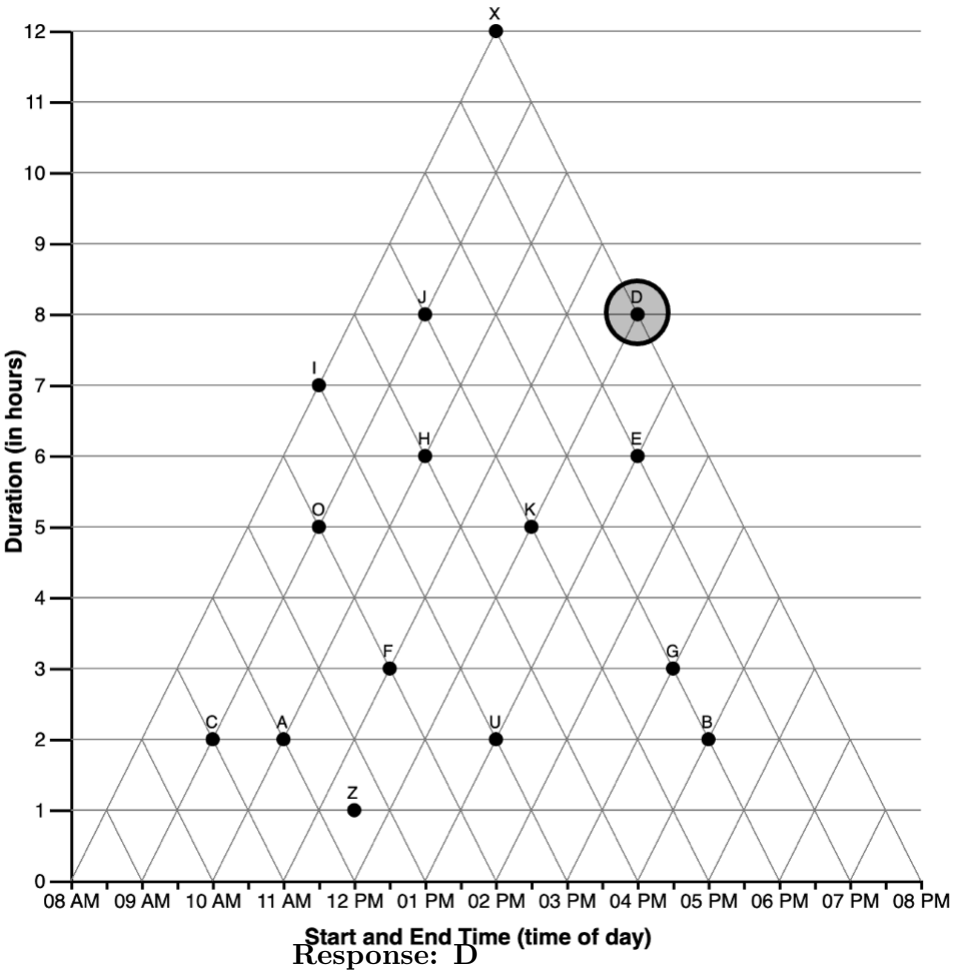
Response: AK

- 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.



Response: J

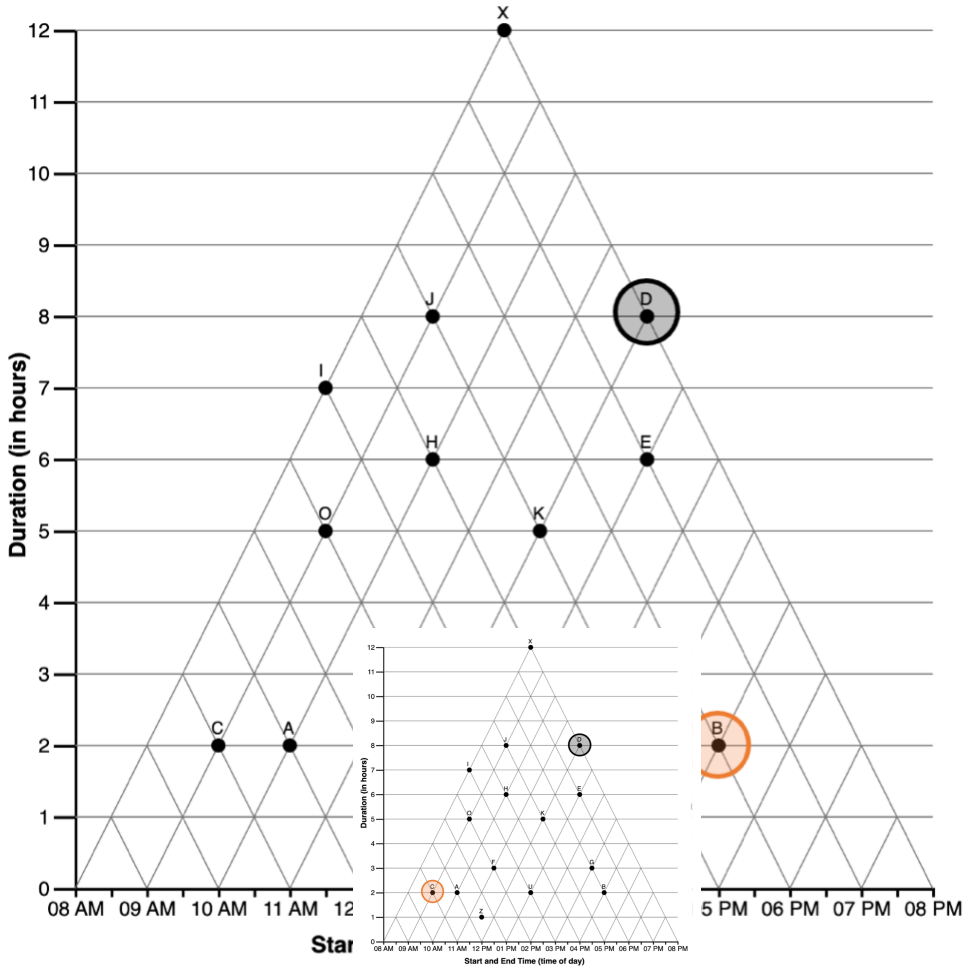
- indicates an **Tversky** strategy following connecting lines
- Consistent with the reader identifying the reference point (D) on the graph, and following its horizontal gridline to the y-axis, locating data point J.



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



2.3.1.2.2 Q2. Impasse Condition

```
q <- keys_raw %>% filter(condition == 121) %>% filter(Q==2)
ignore <- q %>% select("REF_POINT")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL", "SATISFICE_left", "SATISFICE_right","T
ves <- q %>% mutate(
  SATISFICE_left_allow = "",
  SATISFICE_right_allow = ""
) %>% select("TRI_allow", "ORTH_allow", "SATISFICE_left_allow","SATISFICE_right_allow", "T
options <- q %>% select("OPTIONS")
```

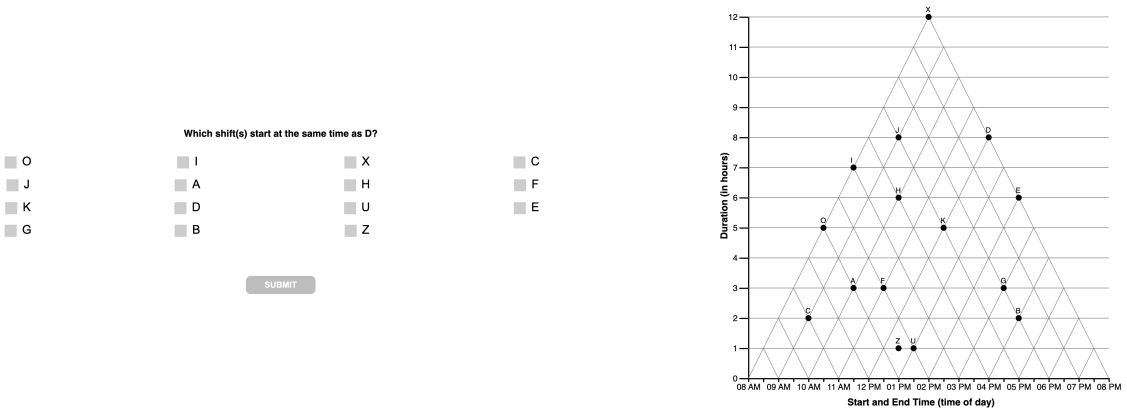


Figure 2.6: Q2—Impasse Condition

```
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Satisficing [left]", "Satisficing [right]", "Tvers
            "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 Impasse 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
```

```
title <- "Frequency of Selected Response Options for Question #2 (Impasse Condition)"
names = c("response","n","interpretation","absolute","tri","tversky","satisfice","orthogon

df_items %>% filter(q == 2 & condition == 121) %>% group_by(response) %>%
  dplyr::summarise( count = n(),
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
                    orthogonal =  unique(score_ORTH),
                    satisficing =  unique(score_SATISFICE),
                    tversky = unique(score_TVERSKY),
                    interpretation = unique(interpretation),
```

Table 2.13: Answer Key | Q2 Impasse Condition : Which shift(s) start at the same time as D?

interpretation	answer	not penalized
Triangular	K	Z
Orthgonal		
Satisficing [left]		
Satisficing [right]	G	
Tversky [maximal]	JKE	Z
Tversky [start diagonal]	K	Z
Tversky [end diagonal]	E	
Tversky [duration line]	J	

Note: 15 response options: AIKGXJDBCHUZOFE

```
                  scaled = unique(score_SCALED)) %>%
  arrange(interpretation, desc(count)) %>%
  select(response, count, interpretation, nice,
         triangular, tversky, satisficing, orthogonal, scaled) %>%
  kbl(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Score" = 1, "Interpretation Scores"=4, "Discriminant
  pack_rows("Triangular", 1, 2) %>%
  pack_rows("Lines-Connect", 3, 10) %>%
  pack_rows("Satisfice", 11, 12) %>%
  pack_rows("Other", 13, 16)  %>%
  pack_rows("Unknown", 17, 18)  %>%
  footnote(general = "n = number of responses in sample",
           general_title = "Note: ",footnote_as_chunk = T)
```

\begin{table}

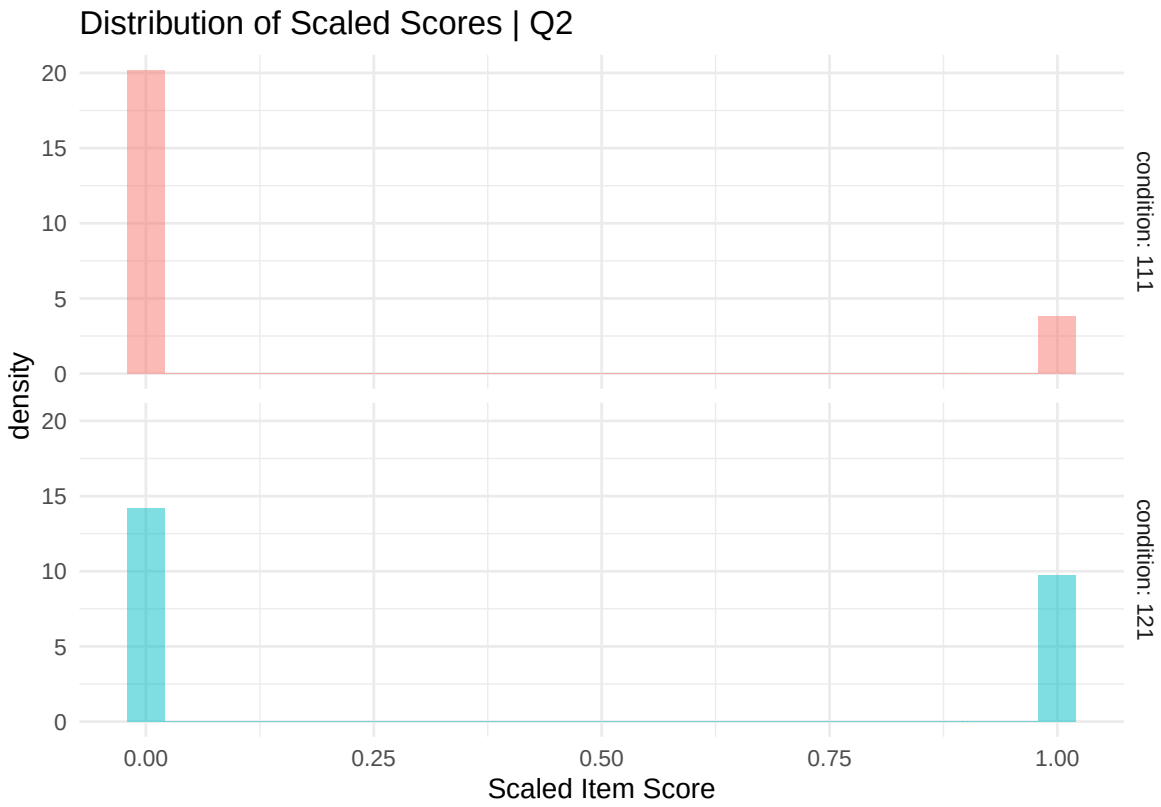
\caption{Frequency of Selected Response Options for Question #2 (Impasse Condition)}

			Strict Score	Interpretation Scores				Discrimi
response	n	interpretation	absolute	tri	tversky	satisfice	orthogonal	scaled s
Triangular								
K	69	Triangular	1	1.000	1.000	-0.077	NA	
DK	1	Triangular	1	1.000	1.000	-0.077	NA	
Lines-Connect								
J	12	Tversky	0	-0.083	1.000	-0.077	NA	
EK	3	Tversky	0	0.917	0.923	-0.154	NA	
EX	2	Tversky	0	-0.167	0.923	-0.154	NA	
BEG	1	Tversky	0	-0.250	0.846	0.846	NA	
E	1	Tversky	0	-0.083	1.000	-0.077	NA	
EKX	1	Tversky	0	0.833	0.846	-0.231	NA	
HJZ	1	Tversky	0	-0.167	0.846	-0.231	NA	
JK	1	Tversky	0	0.917	0.923	-0.154	NA	
Satisfice								
G	19	Satisfice	0	-0.083	-0.077	1.000	NA	
BG	2	Satisfice	0	-0.167	-0.154	0.923	NA	
Other								
D	7	reference	0	0.000	NA	0.000	NA	
	43	blank	0	0.000	NA	0.000	NA	
ACDFHIJKOUXZ	1	frenzy	0	0.250	0.250	-0.846	NA	
BEGKUZ	1	frenzy	0	0.667	0.667	0.615	NA	
Unknown								
C	6	?	0	-0.083	-0.077	-0.077	NA	
FO	1	?	0	-0.167	-0.154	-0.154	NA	

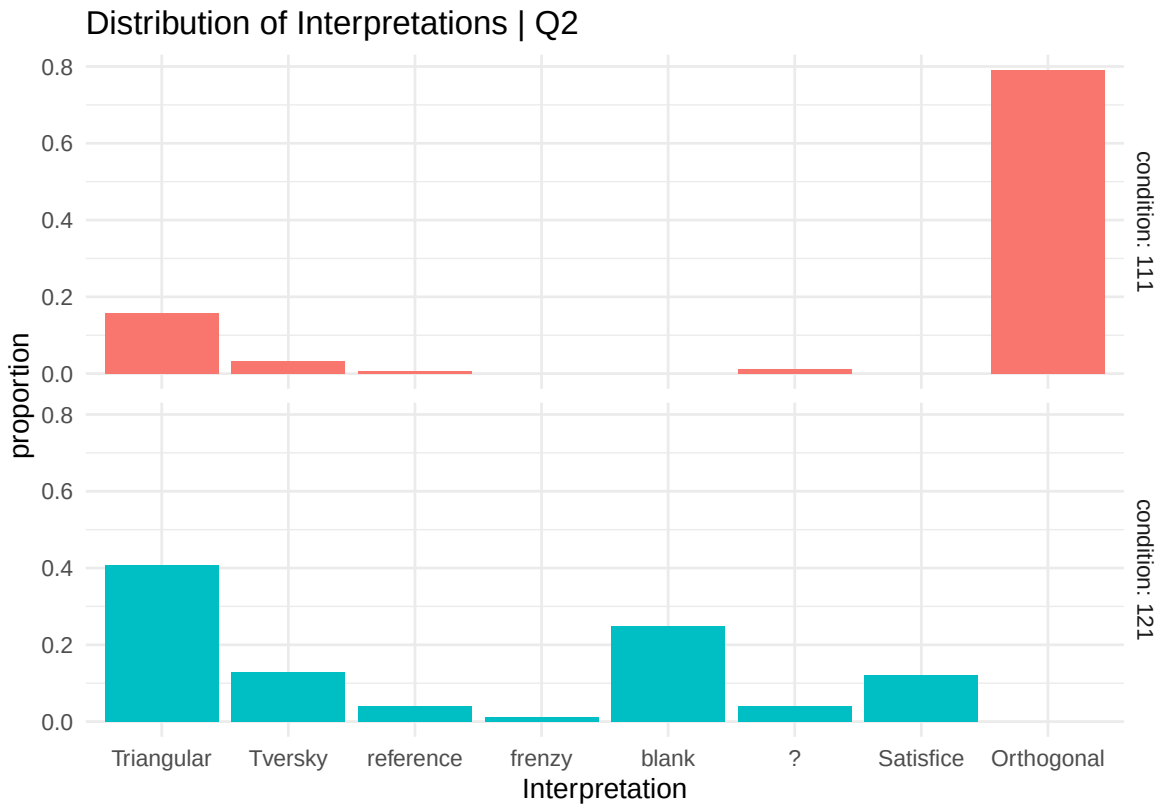
Note: n = number of responses in sample

\end{table}

```
gf_dhistogram(~ score_niceABS, fill = ~condition, data = df_items %>% filter(q ==2)) %>%
  gf_facet_grid( condition ~ ., labeller = label_both) +
  labs( x = "Scaled Item Score", title = "Distribution of Scaled Scores | Q2 ") +
  theme_minimal() + theme(legend.position = "blank")
```




```
gf_props(~sorted_interpretation, fill = ~condition, data = df_items %>% filter(q ==2)) %>%
  gf_facet_grid( condition ~ ., labeller = label_both) +
  labs( x = "Interpretation", title = "Distribution of Interpretations | Q2 ") +
  theme_minimal() + theme(legend.position = "blank")
```



2.3.1.3 Question #3

2.3.1.3.1 Q3. Control Condition

```
q <- keys_raw %>% filter(condition == 111) %>% filter(Q==3)
ignore <- q %>% select("REF_POINT")
answers <- q %>% select("TRIANGULAR", "ORTHOGONAL", "SATISFICE_left", "SATISFICE_right","T
ves <- q %>% mutate(
  SATISFICE_left_allow = "",
  SATISFICE_right_allow = ""
) %>% select("TRI_allow", "ORTH_allow", "SATISFICE_left_allow","SATISFICE_right_allow", "T
options <- q %>% select("OPTIONS")
question = q %>%  select("TEXT")
scores <- c("Triangular", "Orthgonal", "Satisficing [left]", "Satisficing [right]", "Tvers
            "Tversky [end diagonal]", "Tversky [duration line]")
d = tibble(interpretation = scores, answer = answers, allowed=ves)
d$answer <- replace_na(d$answer, "")
```

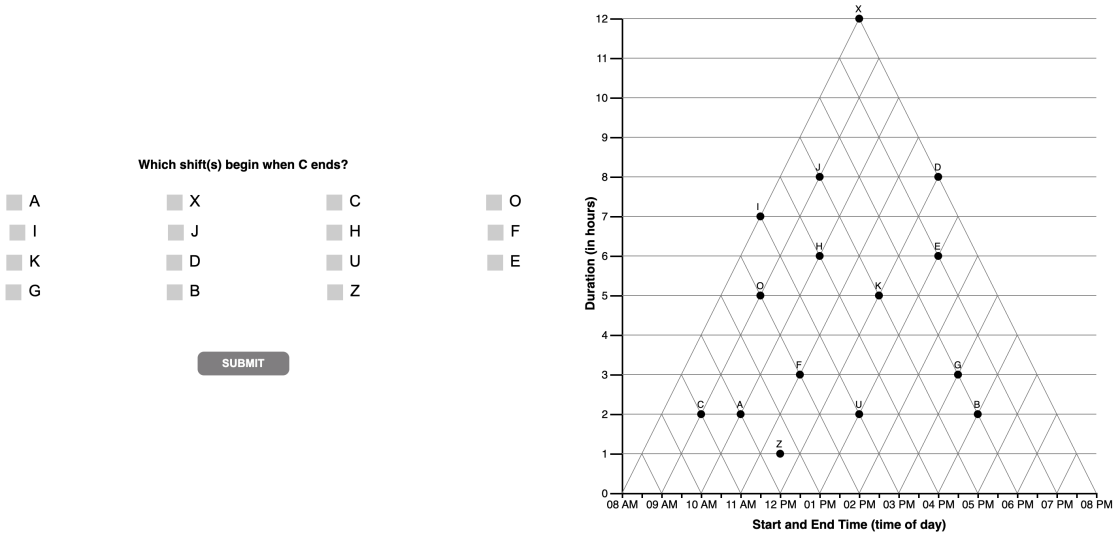


Figure 2.7: Q3—Control Condition

```
d$allowed <- replace_na(d$allowed, "")

title = paste("Answer Key | Q3 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
```

```
title <- "Frequency of Selected Response Options for Question #3 (Control Condition)"
names = c("response","n","interpretation","absolute","tri","tversky","satisfice","orthogon

df_items %>% filter(q == 3 & condition == 111) %>% group_by(response) %>%
  dplyr::summarise( count = n(),
                    nice = unique(score_niceABS),
                    triangular = unique(score_TRI),
                    orthogonal =  unique(score_ORTH),
                    satisficing =  unique(score_SATISFICE),
                    tversky = unique(score_TVERSKY),
                    interpretation = unique(interpretation),
```

Table 2.14: Answer Key | Q3 Control Condition : Which shift(s) begin when C ends?

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

Note: 15 response options: AIKGXJDBCHUZOFE

```
                scaled = unique(score_SCALED)) %>%
  arrange(interpretation, desc(count)) %>%
  select(response, count, interpretation, nice,
         triangular, tversky, satisficing, orthogonal, scaled) %>%
  kbl(caption = title, col.names = names) %>%  kable_classic() %>%
  add_header_above(c(" " = 3, "Strict Score" = 1, "Interpretation Scores"=4, "Discriminant
  pack_rows("Triangular", 1, 2) %>%
  pack_rows("Lines-Connect", 3, 7) %>%
  pack_rows("Orthogonal", 8, 8) %>%
  pack_rows("Other", 9, 10) %>%
  pack_rows("Unknown", 11, 17)
```

\begin{table}

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