SGC-X

COMBINED | 3 Description

Amy Rae Fox

Table of contents

Part I

MAIN

SGC 3 — The Insight Hypothesis

SGC 4 — The Graphical Framework

SGC 5 — The Effectiveness of Interaction

Scoring Strategy

The purpose of this notebook is to describe the strategy for assigning a score (a measure of accuracy) to response data for the SGC studies. 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.)

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

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

MULTIPLE RESPONSE SCORING

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

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

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

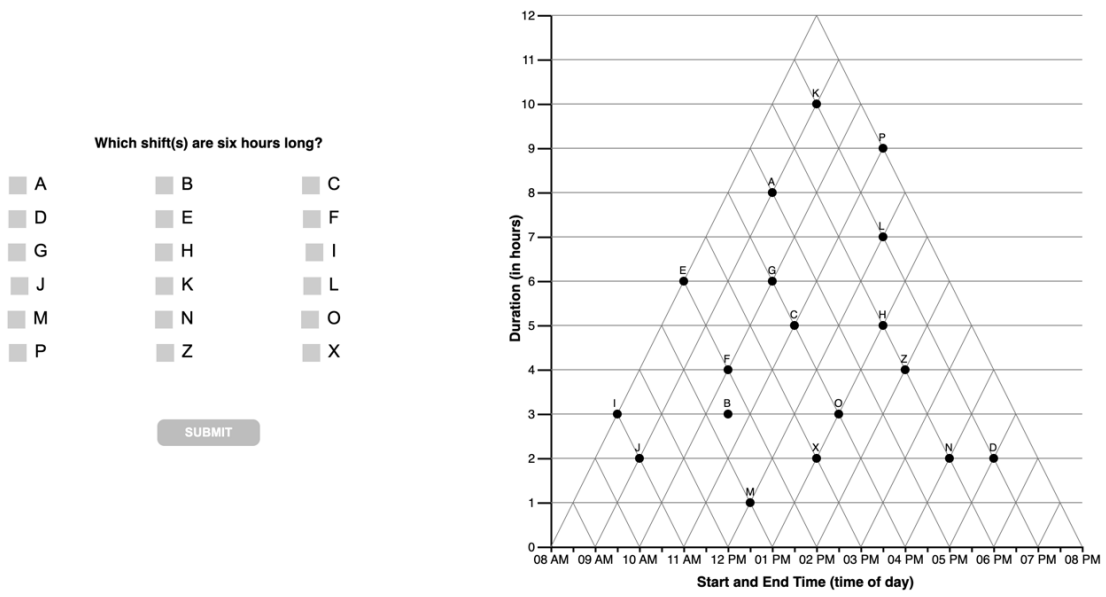


Figure 1: **Figure 1. Sample Graph Comprehension (Question # 6)**

(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.

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

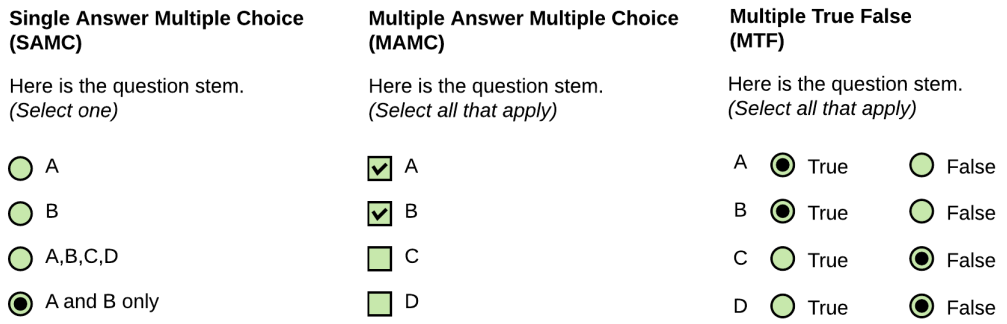


Figure 2: **Figure 2. SAMC (vs) MAMC (vs) MTF**

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].

Scoring Schemes

In the sections that follow, we use the terminology:

Properties of the Stimulus-Question

$$n = \text{number of response options} \tag{0.1}$$
$$= p + q \tag{0.2}$$
$$p = \text{number of true-select options (i.e. should be selected)} \tag{0.3}$$
$$q = \text{number of true-unselect options (i.e. should not be selected)} \tag{0.4}$$

Properties of the Subject's Response

$$i = \text{number of options in correct state}, (0 \leq i \leq n) \tag{0.5}$$
$$f = \text{resulting score} \tag{0.6}$$

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))

}
```

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:

$$f = (1/n * i) - (1/n * (n - i)) \tag{0.7}$$
$$= (2i - n)/n \tag{0.8}$$

```
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)
}
```

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

$$p = \text{number of true-select options (i.e. should be selected)} \tag{0.9}$$
$$q = \text{number of true-unselect options (i.e. should not be selected)} \tag{0.10}$$
$$n = \text{number of options } (n = p + q) \tag{0.11}$$

Properties of Response

$$p_s = \text{number of true-select options selected (i.e. number of correctly checked options)} \tag{0.12}$$
$$q_s = \text{number of true-unselect options selected (i.e. number of incorrectly checked options} \tag{0.13}$$

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

$$f = (p_s/p) - (q_s/q) \tag{0.14}$$
$$\tag{0.15}$$

```
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)), "")
}
```

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)
```

Table 1: 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

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

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

- We see that in the Dichotomous scheme, only the precisely correct response (row 1) yields a score other than zero. This scheme does now allow us to differentiate between different response patters.
- The Partial $[-1/n, +1/n]$ scheme yields a range from $[-1, 1]$, differentiating between responses. However, a blank response (bottom row) receives the same score (0.6) as the selection of the correct option + 1 incorrect option (row 2), which is problematic with

for the goals of this study, where we need to differentiate between states of confusion or uncertainty yielding blank responses and the intentional selection of incorrect items.

- The Partial $[-1/q, +1/p]$ scheme also yields a range of scores from $[-1, 1]$. A blank response (bottom row) yields the same score (0) as the selection of *all* answer options (row 7); both are patterns of behavior we would expect to see if a respondent is confused or uncertain that there is a correct answer to the question.

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

```
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__..._O",
              "AB_..._O",
              "___..._O",
              "___...NO",
              "_BC...NO",
              "ABC...NO",
              "___..._O" )

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

```r
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() %>%
```

Table 2: 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

```
    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.

SCORING 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:

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.

Based on prior observational studies where we observed emergence of other alternative interpretations (i.e. transitional interpretations) we also calculate subscores for these alternatives.

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 strategy) answer key.

For each study in the SGC project, MR data will be scored by following these steps:

(1) Preparing answer keys: For each dataset+question set combination, an answer key is that defines the 'correct' answer set under *each* interpretation of the graph (i.e. a triangular answer, an orthogonal answer, etc).

(2) Calculate strategy scores: Using the strategy specific answer keys, an interpretation subscore is calculated for each response for each interpretation.

(3) Interpretation classification: The interpretation subscores are compared in order to classify each response as a particular interpretation. If no classification can be made, the response is classified as '?'.

(4) Calculate Absolute and Scaled Scores: Two final scores are calculated for each response; an Absolute score that indicates if the response was precisely correct according to the triangular interpretation, and a Scaled score that assigns a numeric value to the interpretation given by the response (ranging from -1 to +1)

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.

```
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# setwd(imac)

#LOAD INDIVIDUAL KEY FILES
key_111_raw <- read_csv('analysis/utils/keys/SGCX_scaffold_111_key.csv') %>% mutate(condit
key_121_raw <- read_csv('analysis/utils/keys/SGCX_scaffold_121_key.csv')%>% mutate(conditi
cs = rep('c', 23) %>% str_c(collapse="") #create column spec
key_test_raw <- read_csv('analysis/utils/keys/SGCX_test_key.csv', col_types = cs)%>% mutat

#JOIN THEM
keys_raw <- rbind(key_111_raw, key_121_raw, key_test_raw )

#CLEANUP
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.

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

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_tri <- keys_tri %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
)

#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

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,

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_orth <- keys_orth %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
)

#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

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,

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_tversky_max <- keys_tversky_max %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
)

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

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_tversky_start <- keys_tversky_start %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
)

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

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_tversky_end <- keys_tversky_end %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
)

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_

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_tversky_duration <- keys_tversky_duration %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
)

#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

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. The "Satisficing" strategy involves the reader selecting data points *nearest* to the orthogonal projection from the reference point in the question. We observe this strategy in some readers when there is no orthogonal response available (i.e. in the impasse condition), so they select the points nearest to the projection (i.e. "close enough").

```
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,

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_satisfice_right <- keys_satisfice_right %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
)

#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

#4. replace condition 111 with "general" to accomodate other conditions [only 121 is speci
keys_satisfice_left <- keys_satisfice_left %>% mutate(
  condition = replace(condition, condition != "121", "DEFAULT")
```

```
)

#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

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

Finally, we need to clean up and generalize our answer keys to accommodate the experimental conditions for Study SGC4-SGC5. In both of these studies the answer set (and underlying graphed data set) are identical, the conditions differ only based on the structure of the gridlines or marks used to represent the data, or interactive mode of the answer format.

```
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# setwd(imac)

#SAVE KEYS FOR FUTURE USE
write.csv(keys_raw,"analysis/utils/keys/parsed_keys/keys_raw", row.names = FALSE)
write.csv(keys_orth,"analysis/utils/keys/parsed_keys/keys_orth", row.names = FALSE)
write.csv(keys_tri,"analysis/utils/keys/parsed_keys/keys_tri", row.names = FALSE)
write.csv(keys_satisfice_left,"analysis/utils/keys/parsed_keys/keys_satisfice_left", row.n
write.csv(keys_satisfice_right,"analysis/utils/keys/parsed_keys/keys_satisfice_right", row
write.csv(keys_tversky_duration,"analysis/utils/keys/parsed_keys/keys_tversky_duration", r
write.csv(keys_tversky_end,"analysis/utils/keys/parsed_keys/keys_tversky_end", row.names =
write.csv(keys_tversky_max,"analysis/utils/keys/parsed_keys/keys_tversky_max", row.names =
write.csv(keys_tversky_start,"analysis/utils/keys/parsed_keys/keys_tversky_start", row.nam
```

2. Calculate Subscores

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** interpretation?
  - `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). This is a more generous version of the Absolute score that does not penalize the participant if in addition to the correct answer *in addition to* they also select the data point referenced in the question.

To facilitate scoring, we import the following helper functions in each scoring script.

```
# #HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# setwd(imac)
```

```r
source("analysis/utils/scoring.R")

print(calc_subscore)
```

```
function (question, cond, response, keyframe)
{
    if (cond == 121 & question < 6) {
        p = keyframe %>% filter(Q == question) %>% filter(condition ==
            "121") %>% select(set_p) %>% pull(set_p) %>% str_split("") %>%
            unlist()
        q = keyframe %>% filter(Q == question) %>% filter(condition ==
            "121") %>% select(set_q) %>% pull(set_q) %>% str_split("") %>%
            unlist()
        pn = keyframe %>% filter(Q == question) %>% filter(condition ==
            "121") %>% select(n_p)
        qn = keyframe %>% filter(Q == question) %>% filter(condition ==
            "121") %>% select(n_q)
    }
    else {
        p = keyframe %>% filter(Q == question) %>% filter(condition ==
            "DEFAULT") %>% select(set_p) %>% pull(set_p) %>%
            str_split("") %>% unlist()
        q = keyframe %>% filter(Q == question) %>% filter(condition ==
            "DEFAULT") %>% select(set_q) %>% pull(set_q) %>%
            str_split("") %>% unlist()
        pn = keyframe %>% filter(Q == question) %>% filter(condition ==
            "DEFAULT") %>% select(n_p)
        qn = keyframe %>% filter(Q == question) %>% filter(condition ==
            "DEFAULT") %>% select(n_q)
    }
    if (response != "") {
        response = response %>% str_split("") %>% unlist()
    }
    ps = length(intersect(response, p))
    qs = length(intersect(response, q))
    x = f_partialP(ps, pn, qs, qn) %>% unlist() %>% as.numeric()
    rm(p, q, pn, qn, ps, qs)
    return(x)
}
```

3. Derive Interpretation

Next, we use the interpretation subscores to classify the response as a particular interpretation. This classification algorithm : (1) First decides if the response matches one or more 'special' situations (blank response, reference point response, both ORTH and TRI) (2) If response doesn't match a special situation, it compares the individual subscores, and subscores and decides if they are *discriminant* (i.e. are the scores different enough to make a prediction). A discriminant threshold of 0.5pts (on a scale from -1 to +1 is used) (2) If the variance in subscores surpasses the threshold, the interpretation is classified based on the highest subscore (TRIANGULAR, ORTHOGONAL, TVERSKY, SATISFICE) (3) If the variance does not surpass the threshold, the interpretation is labelled as "?", indicating it cannot be classified, and is of an unknown interpretation.

The final output is called `interpretation`.

```
print(derive_interpretation)
```

```
function (df)
{
    threshold_range = 0.5
    threshold_frenzy = 4
    for (x in 1:nrow(df)) {
        t = df[x, ] %>% dplyr::select(score_TV_max, score_TV_start,
            score_TV_end, score_TV_duration)
        t.long = gather(t, score, value, 1:4)
        t.long[t.long == ""] = NA
        if (length(unique(t.long$value)) == 1) {
            if (is.na(unique(t.long$value))) {
                df[x, "score_TVERSKY"] = NA
                df[x, "tv_type"] = NA
            }
        }
        else {
            df[x, "score_TVERSKY"] = as.numeric(max(t.long$value,
                na.rm = TRUE))
            df[x, "tv_type"] = t.long[which.max(t.long$value),
                "score"]
        }
        t = df[x, ] %>% dplyr::select(score_SAT_left, score_SAT_right)
        t.long = gather(t, score, value, 1:2)
        t.long[t.long == ""] = NA
        if (length(unique(t.long$value)) == 1) {
```

```
    if (is.na(unique(t.long$value))) {
        df[x, "score_SATISFICE"] = NA
        df[x, "sat_type"] = NA
    }
}
else {
    df[x, "score_SATISFICE"] = as.numeric(max(t.long$value,
        na.rm = TRUE))
    df[x, "sat_type"] = t.long[which.max(t.long$value),
        "score"]
}
t = df[x, ] %>% dplyr::select(score_TRI, score_TVERSKY,
    score_SATISFICE, score_ORTH)
t.long = gather(t, score, value, 1:4)
t.long[t.long == ""] = NA
df[x, "top_score"] = as.numeric(max(t.long$value, na.rm = TRUE))
df[x, "top_type"] = t.long[which.max(t.long$value), "score"]
r = as.numeric(range(t.long$value, na.rm = TRUE))
r = diff(r)
df[x, "range"] = r
if (r < threshold_range) {
    df[x, "best"] = "?"
}
else {
    p = df[x, "top_type"]
    if (p == "score_TRI") {
        df[x, "best"] = "Triangular"
    }
    else if (p == "score_ORTH") {
        df[x, "best"] = "Orthogonal"
    }
    else if (p == "score_TVERSKY") {
        df[x, "best"] = "Tversky"
    }
    else if (p == "score_SATISFICE") {
        df[x, "best"] = "Satisfice"
    }
}
if (!is.na(df[x, "score_BOTH"])) {
    if (df[x, "score_BOTH"] == 1) {
        df[x, "best"] = "both tri + orth"
    }
}
```

```
        if (df[x, "num_o"] == 0) {
            df[x, "best"] = "blank"
        }
        if (df[x, "num_o"] > threshold_frenzy) {
            df[x, "best"] = "frenzy"
        }
        if (!is.na(df[x, "score_REF"])) {
            if (df[x, "score_REF"] == 1) {
                df[x, "best"] = "reference"
            }
        }
    }
    rm(t, t.long, x, r, p)
    rm(threshold_frenzy, threshold_range)
    df$int2 <- factor(df$best, levels = c("Triangular", "Tversky",
        "Satisfice", "Orthogonal", "reference", "both tri + orth",
        "blank", "frenzy", "?"))
    df$interpretation <- factor(df$best, levels = c("Orthogonal",
        "Satisfice", "frenzy", "?", "reference", "blank", "both tri + orth",
        "Tversky", "Triangular"))
    df$high_interpretation <- fct_collapse(df$interpretation,
        orthogonal = c("Satisfice", "Orthogonal"), neg.trans = c("frenzy",
            "?"), neutral = c("reference", "blank"), pos.trans = c("Tversky",
            "both tri + orth"), triangular = "Triangular")
    df$tv_type = as.factor(df$tv_type)
    df$top_type = as.factor(df$top_type)
    df$high_interpretation = factor(df$high_interpretation, levels = c("orthogonal",
        "neg.trans", "neutral", "pos.trans", "triangular"))
    df <- df %>% dplyr::select(-best)
    return(df)
}
```

4. Derive Scaled Score

The `interpretation` response variable gives us the finest grain indication of the reader's understanding of the graph for a particular question. However, it is a categorical variable, which poses a challenge for analyzing cumulative performance at the subject level. To address this challenge, we derive a *scaled_score* that converts each possible interpretation to a numeric value on a scale from -1 to +1. This scaling takes advantage of the observation that each interpretation can be positioned along a spectrum of understanding from completely incorrect (orthogonal) to completely correct (triangular). Alternative interpretations lay somewhere between.

Specifically, we assign the following values to each interpretation:

- (-1) : ORTHOGONAL, SATISFICE (satisfice represents an attempt at an orthogonal answer when none is available)
- (-0.5): ? (some alternative that cannot be identified; but meaningful that it is not orthogonal)
- (0): REFERENCE POINT, BLANK (indicates the individual thinks there is no answer, recognizes that ORTHOGONAL cannot be correct, but does not conceive of triangular)
- (+0.5) TVERSKY, BOTH TRI + ORTH (indicates that they "see" a triangular response, but lack certainty and also select the ORTHOGONAL response)
- (+1) TRIANGULAR +1

```
print(calc_scaled)
```

```
function (v)
{
    v <- recode(v, Orthogonal = -1, Satisfice = -1, frenzy = -0.5,
        `?` = -0.5, reference = 0, blank = 0, `both tri + orth` = 0.5,
        Tversky = 0.5, Triangular = 1)
    return(v)
}
```

5. Summarize by Subject

The final step in the scoring procedure is to summarise the item-level scores by subject, and save certain summaries to the subject-level record. We also construct two long-format dataframes containing cummulative progress scores (the point-in-time [absolute, scaled] scores for each subject on each question).

```
print(summarise_bySubject)
```

```
function (subjects, items)
{
    subjects_summary <- items %>% filter(q %nin% c(6, 9)) %>%
        group_by(subject) %>% dplyr::summarise(subject = as.character(subject),
        s_TRI = sum(score_TRI, na.rm = TRUE), s_ORTH = sum(score_ORTH,
            na.rm = TRUE), s_TVERSKY = sum(score_TVERSKY, na.rm = TRUE),
        s_SATISFICE = sum(score_SATISFICE, na.rm = TRUE), s_REF = sum(score_REF,
            na.rm = TRUE), s_ABS = sum(score_ABS, na.rm = TRUE),
        s_NABS = sum(score_niceABS, na.rm = TRUE), s_SCALED = sum(score_SCALED,
```

```
        na.rm = TRUE), DV_percent_NABS = s_NABS/13, rt_m = sum(rt_s)/60,
    item_avg_rt = mean(rt_s), item_min_rt = min(rt_s), item_max_rt = max(rt_s),
    item_n_TRI = sum(interpretation == "Triangular"), item_n_ORTH = sum(interpretation ==
        "Orthogonal"), item_n_TV = sum(interpretation ==
        "Tversky"), item_n_SAT = sum(interpretation == "Satisfice"),
    item_n_OTHER = sum(interpretation %nin% c("Triangular",
        "Orthogonal", "Tversky", "Satisfice")), item_n_POS = sum(high_interpretation ==
        "pos.trans"), item_n_NEG = sum(high_interpretation ==
        "neg.trans"), item_n_NEUTRAL = sum(high_interpretation ==
        "neutral")) %>% arrange(subject) %>% slice(1L)
subjects_q1 <- items %>% filter(q == 1) %>% mutate(item_q1_NABS = score_niceABS,
    item_q1_SCALED = score_SCALED, item_q1_interpretation = interpretation,
    item_q1_rt = rt_s, ) %>% dplyr::select(subject, item_q1_NABS,
    item_q1_SCALED, item_q1_interpretation, item_q1_rt) %>%
    arrange(subject)
subjects_q5 <- items %>% filter(q == 5) %>% mutate(item_q5_NABS = score_niceABS,
    item_q5_SCALED = score_SCALED, item_q5_interpretation = interpretation,
    item_q5_rt = rt_s, ) %>% dplyr::select(subject, item_q5_NABS,
    item_q5_SCALED, item_q5_interpretation, item_q5_rt) %>%
    arrange(subject)
subjects_q7 <- items %>% filter(q == 7) %>% mutate(item_q7_NABS = score_niceABS,
    item_q7_interpretation = interpretation, item_q7_rt = rt_s,
    ) %>% dplyr::select(subject, item_q7_NABS, item_q7_interpretation,
    item_q7_rt) %>% arrange(subject)
subjects_q15 <- items %>% filter(q == 15) %>% mutate(item_q15_NABS = score_niceABS,
    item_q15_interpretation = interpretation, item_q15_rt = rt_s,
    ) %>% dplyr::select(subject, item_q15_NABS, item_q15_interpretation,
    item_q15_rt) %>% arrange(subject)
subjects_scaffold <- items %>% filter(q < 6) %>% group_by(subject) %>%
    dplyr::summarise(item_scaffold_NABS = sum(score_niceABS),
        item_scaffold_SCALED = sum(score_SCALED), item_scaffold_rt = sum(rt_s)/60) %>%
    dplyr::select(subject, item_scaffold_NABS, item_scaffold_SCALED,
        item_scaffold_rt) %>% arrange(subject)
subjects_test <- items %>% filter(q %nin% c(1, 2, 3, 4, 5,
    6, 9)) %>% group_by(subject) %>% dplyr::summarise(item_test_NABS = sum(score_niceABS)
    item_test_SCALED = sum(score_SCALED), item_test_rt = sum(rt_s)/60) %>%
    dplyr::select(subject, item_test_NABS, item_test_SCALED,
        item_test_rt) %>% arrange(subject)
print(unique(subjects_summary$subject == subjects$subject))
print(unique(subjects_summary$subject == subjects_q1$subject))
print(unique(subjects_summary$subject == subjects_q5$subject))
print(unique(subjects_summary$subject == subjects_q7$subject))
print(unique(subjects_summary$subject == subjects_q15$subject))
```

```
    print(unique(subjects_summary$subject == subjects_scaffold$subject))
    print(unique(subjects_summary$subject == subjects_test$subject))
    x = merge(subjects, subjects_summary, by.x = "subject", by.y = "subject")
    x = merge(x, subjects_q1)
    x = merge(x, subjects_q5)
    x = merge(x, subjects_q7)
    x = merge(x, subjects_q15)
    x = merge(x, subjects_scaffold)
    x = merge(x, subjects_test)
    subjects <- x
    rm(subjects_q1, subjects_q5, subjects_q7, subjects_q15, subjects_scaffold,
        subjects_test, subjects_summary, x)
    return(subjects)
}
```

```
print(progress_Absolute)
```

```
function (items)
{
    x <- items %>% filter(q %nin% c(6, 9)) %>% dplyr::select(subject,
        mode, condition, q, score_niceABS)
    wide <- x %>% pivot_wider(names_from = q, names_glue = "q_{q}",
        values_from = score_niceABS)
    wide$c1 = wide$q_1
    wide$c2 = wide$c1 + wide$q_2
    wide$c3 = wide$c2 + wide$q_3
    wide$c4 = wide$c3 + wide$q_4
    wide$c5 = wide$c4 + wide$q_5
    wide$c6 = wide$c5 + wide$q_7
    wide$c7 = wide$c6 + wide$q_8
    wide$c8 = wide$c7 + wide$q_10
    wide$c9 = wide$c8 + wide$q_11
    wide$c10 = wide$c9 + wide$q_12
    wide$c11 = wide$c10 + wide$q_13
    wide$c12 = wide$c11 + wide$q_14
    wide$c13 = wide$c12 + wide$q_15
    wide <- wide %>% dplyr::select(subject, mode, condition,
        c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13)
    df_absolute_progress <- wide %>% pivot_longer(cols = c1:c13,
        names_to = "question", names_pattern = "c(.*)", values_to = "score")
    df_absolute_progress$question <- as.integer(df_absolute_progress$question)
```

```
    rm(x, wide)
    return(df_absolute_progress)
}
```

```
print(progress_Scaled)
```

```
function (items)
{
    x <- items %>% filter(q %nin% c(6, 9)) %>% select(subject,
        mode, condition, q, score_SCALED)
    wide <- x %>% pivot_wider(names_from = q, names_glue = "q_{q}",
        values_from = score_SCALED)
    wide$c1 = wide$q_1
    wide$c2 = wide$c1 + wide$q_2
    wide$c3 = wide$c2 + wide$q_3
    wide$c4 = wide$c3 + wide$q_4
    wide$c5 = wide$c4 + wide$q_5
    wide$c6 = wide$c5 + wide$q_7
    wide$c7 = wide$c6 + wide$q_8
    wide$c8 = wide$c7 + wide$q_10
    wide$c9 = wide$c8 + wide$q_11
    wide$c10 = wide$c9 + wide$q_12
    wide$c11 = wide$c10 + wide$q_13
    wide$c12 = wide$c11 + wide$q_14
    wide$c13 = wide$c12 + wide$q_15
    wide <- wide %>% select(subject, mode, condition, c1, c2,
        c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13)
    df_scaled_progress <- wide %>% pivot_longer(cols = c1:c13,
        names_to = "question", names_pattern = "c(.*)", values_to = "score")
    df_scaled_progress$question <- as.integer(df_scaled_progress$question)
    rm(x, wide)
    return(df_scaled_progress)
}
```

Modelling Reference

In this notebook we use data from study SGC3A to explore different modelling techniques and assess their suitability for the bimodal accuracy distributions.

Pre-Requisite
2_sgc3A_scoring.qmd

```
library(Hmisc) # %nin% operator

library(ggpubr) #arrange plots
library(cowplot) #arrange shift function plots
library(ggformula) #easy graphs
library(vcd) #mosaic plots
library(vcdExtra) #mosaic plots
library(kableExtra) #printing tables
library(sjPlot) #visualize model coefficients

#plot model estimates with uncertainty
library(ggdist)
library(broom)
library(modelr)
library(distributional)

#models and performance
library(lmerTest) #for CIs in glmer
library(ggstatsplot) #plots w/ embedded stats
library(report) #easystats reporting
library(see) #easystats visualization
library(performance) #easystats model diagnostics
library(qqplotr) #confint on qq plot
library(gmodels) #contingency table and CHISQR
library(equatiomatic) #extract model equation
library(pscl) #zeroinfl / hurdle models
library(lme4) #mixed effects models
```

```r
library(ggeffects) #visualization log regr models

library(tidyverse) #ALL THE THINGS

#OUTPUT OPTIONS
library(dplyr, warn.conflicts = FALSE)
options(dplyr.summarise.inform = FALSE)
options(ggplot2.summarise.inform = FALSE)
options(scipen=1, digits=3)

#GRAPH THEMEING
theme_set(theme_minimal())
```

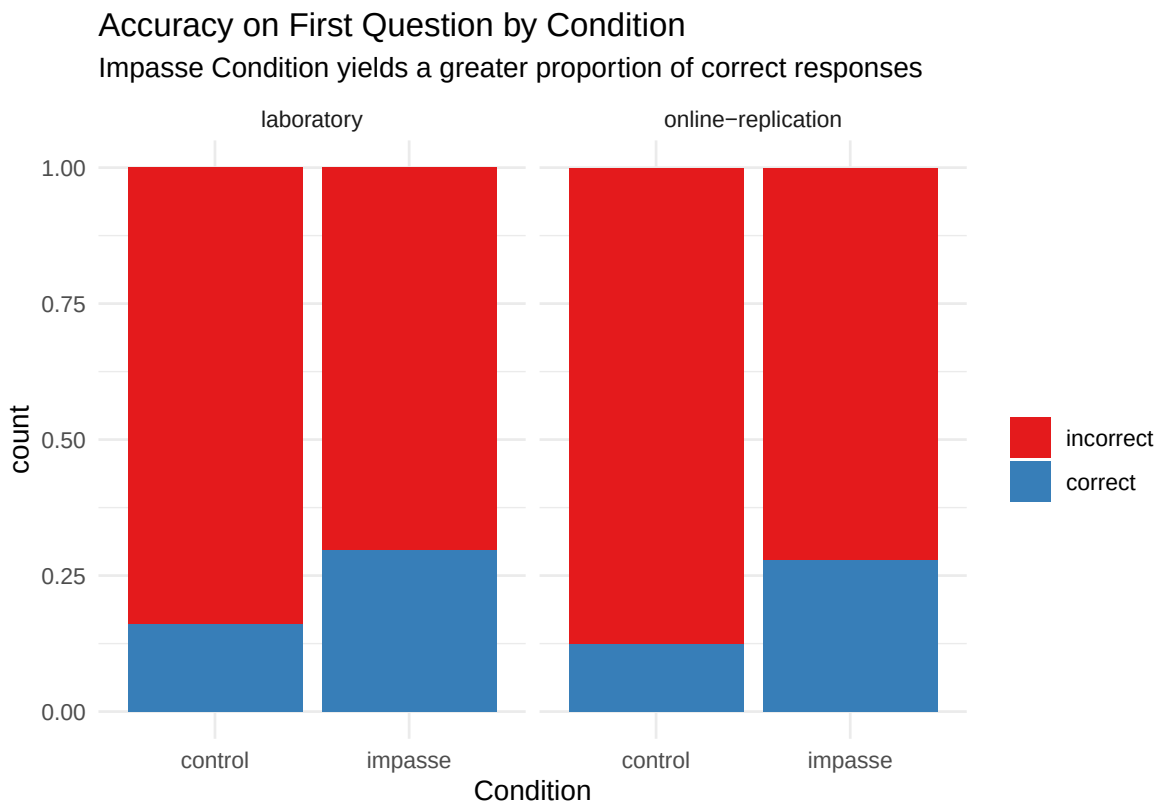
```r
# HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# mbp = "/Users/amyfox/Sites/RESEARCH/SGC-Scaffolding Graph Comprehension/SGC-X/ANALYSIS/M
# setwd(mbp)

#IMPORT DATA
df_items <- read_rds('analysis/SGC3A/data/2-scored-data/sgc3a_scored_items.rds')
df_subjects <- read_rds('analysis/SGC3A/data/2-scored-data/sgc3a_scored_participants.rds')

#PREP DATA
df_lab <- df_subjects %>% filter(pretty_mode == "laboratory")
df_online <- df_subjects %>% filter(pretty_mode == "online-replication")
```

SINGLE ITEM LEVEL

Q1 Absolute, Interpretation Scores

```r
#FILTER THE DATASET [use subjects, bc it has covariates on that record]
df_q1 <- df_subjects %>% mutate(
  accuracy = recode_factor(item_q1_NABS, "0" ="incorrect","1"="correct"),
  rt = item_q1_rt
) %>% dplyr::select(
  accuracy, rt, pretty_condition, pretty_mode
)

#GROUPED PROPORTIONAL BAR CHART
```

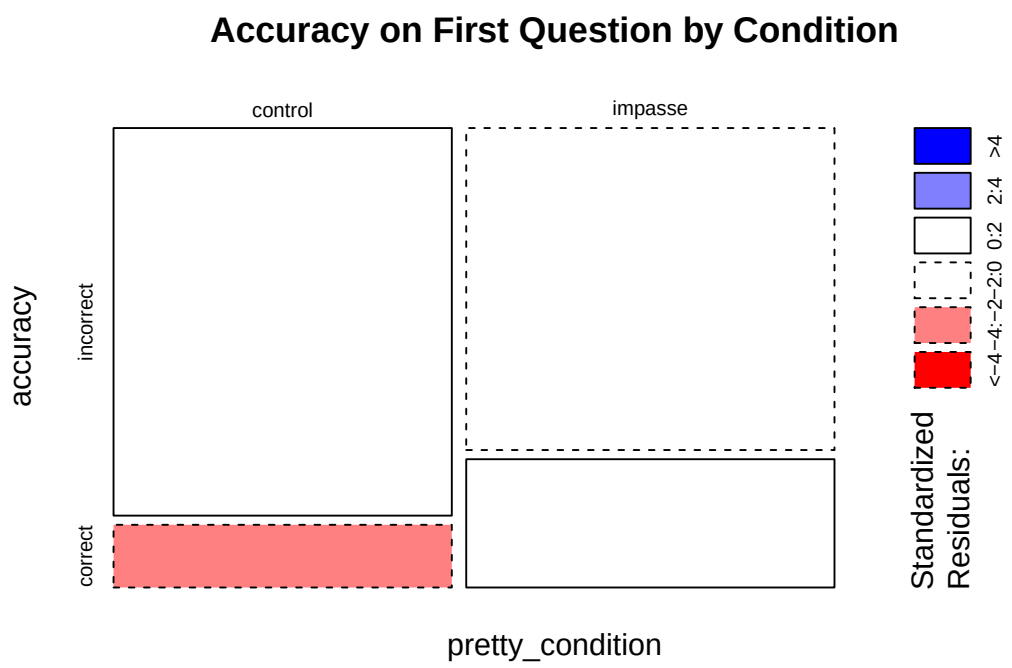
```
# gf_props(~accuracy, fill = ~pretty_condition,
#        position = position_dodge(), data = df_q1) %>%
#   gf_facet_grid(~pretty_mode) +
#    labs(x = "Correct Response on Q 1",
#        title = "Accuracy on First Question by Condition",
#        subtitle="Impasse Condition yields a greater proportion of correct responses") #t

#STACKED BAR CHART
df_q1 %>%
  ggplot(data = .,
         mapping = aes(x = pretty_condition,
                       fill = accuracy)) +
  geom_bar(position = "fill" ) + #,color = "black") +
  scale_fill_brewer(palette = "Set1")  +
  facet_wrap(~pretty_mode) +
   labs(#y = "Correct Response on Q 1",
       title = "Accuracy on First Question by Condition",
       x = "Condition",
       fill = "",
       subtitle="Impasse Condition yields a greater proportion of correct responses")
```



CHI SQUARE

(Combined)

```
#combined dataset
df <- df_q1

#MOSAIC PLOT
#note: blue indicates cell count higher than expected, red indicates cell count less than
mosaicplot(main="Accuracy on First Question by Condition",
           data = df, pretty_condition ~ accuracy, shade = T, color = 2)
```



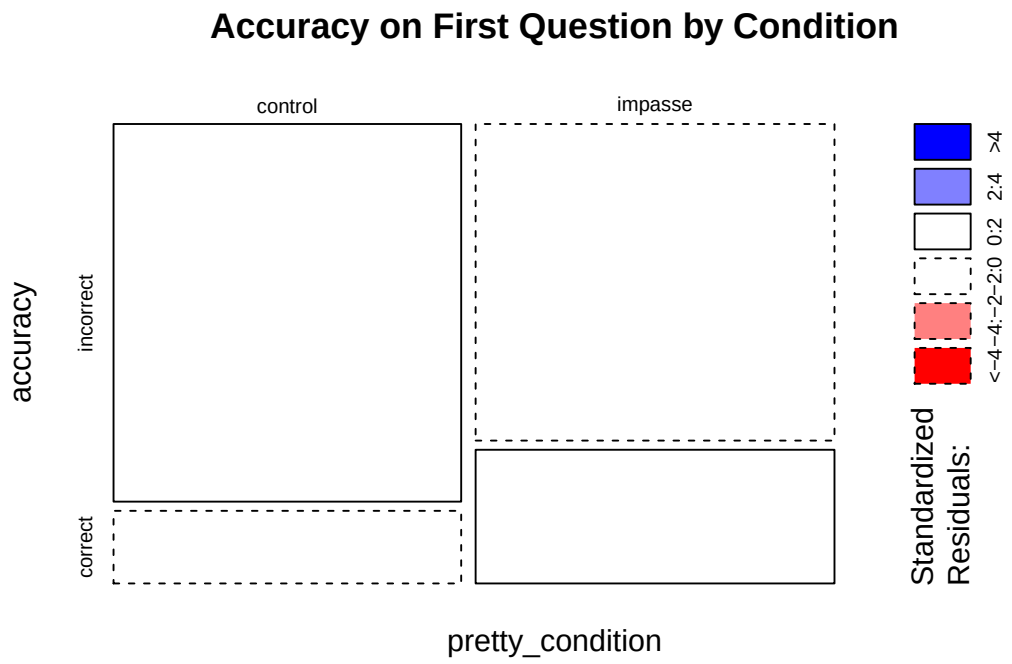

```
# CrossTable( x = df$condition, y = df$accuracy, fisher = TRUE, chisq=TRUE, expected = TRU


df %>%
  sjtab(fun = "xtab", var.labels=c("accuracy", "pretty_condition"),
        show.row.prc=T, show.col.prc=T, show.summary=T, show.exp=T, show.legend=T,
        statistics = "fisher")
```

accuracy

Combining data across both sessions (n=330), a Pearson's Chi-squared test suggests a statistically significant relationship between response accuracy on the first question and experimental condition, χ_2 (1) = 10.3, p = 0.001. The sample odds ratio (2.46, p = 0.001, 95% CI [1.37, 4.53]) indicates that the odds of providing a correct response to the first question are 2.46 higher for subjects in the impasse condition than those in the control condition.

(In Person)

```
#lab only
df <- df_q1 %>% filter(pretty_mode == "laboratory")

#MOSAIC PLOT
#note: blue indicates cell count higher than expected, red indicates cell count less than
mosaicplot(main="Accuracy on First Question by Condition",
            data = df, pretty_condition ~ accuracy,
            shade = T)
```




```
# CrossTable( x = df$condition, y = df$score_niceABS,
#             fisher = TRUE, chisq=TRUE, expected = TRUE, sresid = TRUE)


df %>%
  sjtab(fun = "xtab", var.labels=c("accuracy", "pretty_condition"),
```

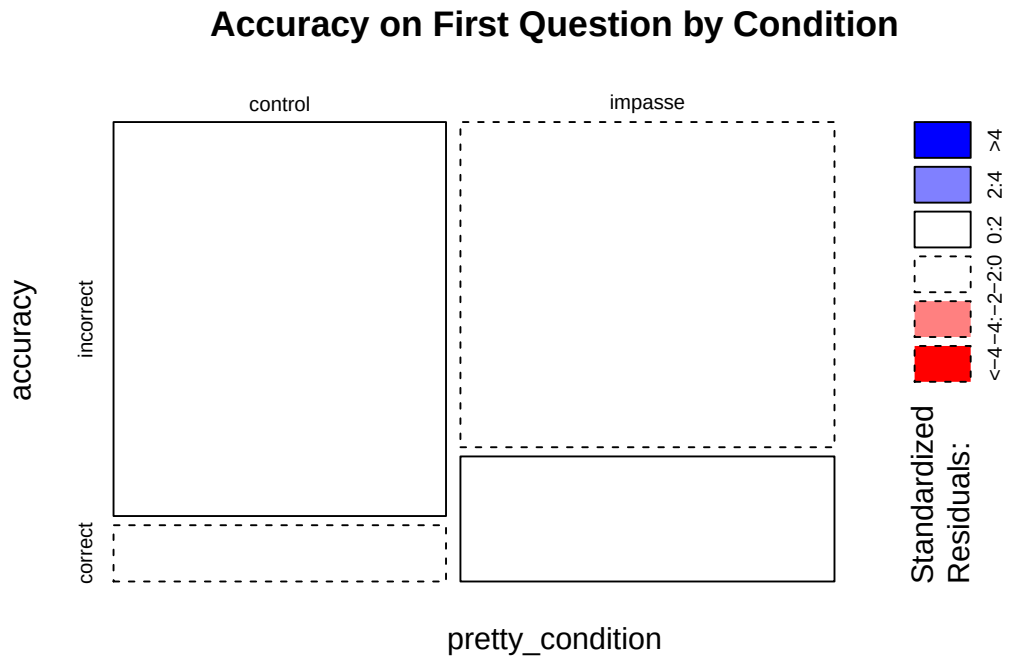
```
        show.row.prc=T, show.col.prc=T, show.summary=T, show.exp=T, show.legend=T,
        statistics = "fisher")
```

accuracy

For (In Person) data collection (n=126) the Pearson's Chi-squared test (of independence) indicates a relationship between response accuracy on the first question and experimental condition that is not significant at the alpha level 0.05, χ^2 (1) = 10.3, p = 0.07. Thus we have insufficient evidence to reject the null hypothesis that the odds ratio is equal to 1. In this particular data sample, the odds ratio (Odds Ratio = 2.18, p = 0.055, 95% CI [0.982, +Inf]) indicates that the odds of producing a correct response on the first question were 2.18 times greater if a subject was in the impasse condition, than in the control condition .

(Online Replication)

```
#online only
df <- df_q1 %>% filter(pretty_mode == "online-replication")

#MOSAIC PLOT
#note: blue indicates cell count higher than expected, red indicates cell count less than
mosaicplot(main="Accuracy on First Question by Condition",
            data = df, pretty_condition ~ accuracy, shade = T)
```




```
# CrossTable( x = df$condition, y = df$score_niceABS, fisher = TRUE,
#             chisq=TRUE, expected = TRUE, sresid = TRUE)


df %>%
  sjtab(fun = "xtab", var.labels=c("accuracy", "pretty_condition"),
        show.row.prc=T, show.col.prc=T, show.summary=T, show.exp=T, show.legend=T,
        statistics = "fisher")
```

accuracy

For online data collection (n=204), a Pearson's Chi-squared test (of independence) indicates a statistically significant relationship between response accuracy on the first question and experimental condition, χ^2 (1) = 7.26, p = 0.009. Thus we have sufficient evidence to reject the null hypothesis that the odds ratio is equal to 1. The odds ratio (Odds Ratio = 2.68, p = 0.005, 95% CI [1.37, +Inf]) indicates that the odds of producing a correct response on the first question were 2.68 times greater if a subject was in the impasse condition, than in the control condition .

LOGISTIC REGRESSION

Fit a logistic regression (at the subject level), predicting Q1 accuracy (absolute score) by condition.

Fit Model

First, we fit a logistic regression with condition as predictor, and compare its fit to an empty (intercept-only) model.

```
#combined
df <- df_items %>% filter(q==1) %>% mutate(
  accuracy = as.factor(score_niceABS)
)

# FREQUENCY TABLE
# my.table <- table(df$accuracy, df$pretty_condition)
# addmargins(my.table) #counts
# addmargins(prop.table(my.table)) #props

# MODEL FITTING::::::::::::::::::::::::::::::::::::

#: 1 EMPTY MODEL baseline glm model intercept only
m0 = glm(accuracy ~ 1, data = df, family = "binomial")
print("EMPTY MODEL")
```

```
[1] "EMPTY MODEL"
```

```
summary(m0)
```

```
Call:
glm(formula = accuracy ~ 1, family = "binomial", data = df)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-0.696  -0.696  -0.696  -0.696   1.753

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.294      0.134   -9.66   <2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 343.66  on 329  degrees of freedom
Residual deviance: 343.66  on 329  degrees of freedom
AIC: 345.7

Number of Fisher Scoring iterations: 4
```

```
#: 2 CONDITION model
m1 <- glm( accuracy ~ pretty_condition, data = df, family = "binomial")
print("PREDICTOR MODEL")
```

```
[1] "PREDICTOR MODEL"
```

```
summary(m1)
```

```
Call:
glm(formula = accuracy ~ pretty_condition, family = "binomial",
    data = df)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-0.819  -0.819  -0.548  -0.548   1.986

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)              -1.822      0.230   -7.93  2.2e-15 ***
pretty_conditionimpasse   0.901      0.285    3.16   0.0016 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 343.66  on 329  degrees of freedom
Residual deviance: 333.07  on 328  degrees of freedom
AIC: 337.1

Number of Fisher Scoring iterations: 4
```

```
#: 3 TEST SUPERIOR FIT
paste("AIC wth predictor is lower than empty model?", m0$aic > m1$aic)
```

```
[1] "AIC wth predictor is lower than empty model? TRUE"
```

```
test_lrt(m0,m1) #same as anova(m0, m1, test = "Chi")
```

```
# Likelihood-Ratio-Test (LRT) for Model Comparison (ML-estimator)

Name | Model | df | df_diff |  Chi2 |     p
-----------------------------------------
m0   |   glm |  1 |         |       |
m1   |   glm |  2 |       1 | 10.59 | 0.001
```

```
paste("Likelihood Ratio test is significant? p = ",(test_lrt(m0,m1))$p[2])
```

```
[1] "Likelihood Ratio test is significant? p =  0.00113745235691825"
```

The Condition predictor significantly improves model fit.

Learning Notes

```
# DESCRIBE MODEL ::::::::::::::::::::::::::::::::::::

print("PREDICTOR MODEL")
```

```
[1] "PREDICTOR MODEL"
```

```
summary(m1)
```

```
Call:
glm(formula = accuracy ~ pretty_condition, family = "binomial",
    data = df)

Deviance Residuals:
```

```
   Min      1Q  Median      3Q     Max
-0.819  -0.819  -0.548  -0.548   1.986

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)               -1.822      0.230   -7.93  2.2e-15 ***
pretty_conditionimpasse    0.901      0.285    3.16   0.0016 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 343.66  on 329  degrees of freedom
Residual deviance: 333.07  on 328  degrees of freedom
AIC: 337.1

Number of Fisher Scoring iterations: 4
```

```
#: INTERPRET COEFFICIENTS

print("Coefficients -- LOG ODDS")
```

```
[1] "Coefficients -- LOG ODDS"
```

```
confint(m1)
```

```
Waiting for profiling to be done...

                         2.5 % 97.5 %
(Intercept)             -2.299  -1.39
pretty_conditionimpasse  0.353   1.48
```

```
print("Coefficients -- ODDS RATIOS")
```

```
[1] "Coefficients -- ODDS RATIOS"
```

```
e <- cbind( exp(coef(m1)), exp(confint(m1))) #exponentiate
```

```
Waiting for profiling to be done...
```

```
e
```

```
                              2.5 % 97.5 %
(Intercept)             0.162  0.10  0.248
pretty_conditionimpasse 2.463  1.42  4.374
```

Understanding the logistic regression model

The logistic regression intercept gives the log odds of the outcome for the reference level of the predictor variable

The logistic regression coefficients give the change in the log odds of the outcome for a one unit increase in the predictor variable.

[the empty model

- The intercept of an empty model (glm(accuracy ~ 1) is equal to log(p/(1-p)), where p = the overall probability of a correct response (df$accuracy ==1).
- In SGC3A Q1 accuracy this = 71 correct / 330 = 0.215 -> log(0.215 / (1-0.215)) = -1.29.
- In other words, the intercept from the model with no predictor variables is the estimated log odds of a correct response for the whole sample.
- We can also transform the log of the odds back to a probability: p = ODDS/ (1+ODDS) = exp(-1.29)/(1+exp(-1.29)) = 0.215. This should matched the prediction of the empty model

[a dichotomous predictor]

natural log (odds of +) = -1.822 + 0.901(x1) ; x1 = 0 for control, 1 for impasse

- INTERCEPT: log odds of (+ response) in control condition
  - log odds of (+) in control : -1.822 + 0.9(0) = -1.822
  - convert to odds by exponentiating the coefficients
    log odds of (+) in control = exp(-1.822) = 0.162 odds
  - convert to probability by formula =>
    p(+) = odds / (1+odds) = 0.162 / (1 + 0.162) = 0.139
    probability of (+) in control = ~14%
- B1 COEFFICIENT: DIFFERENCE in log odds of (+) in impasse vs. control
  - log odds of (+) in impasse: -1.822 + 0.901 = -0.921
  - convert to odds by exponentiating log odds
    log odds (+) in impasse = exp(-0.921) = 0.398
  - convert to probability by formula =>
    p(+) = odds / (1 + odds) = 0.398 / (1+0.398) = 0.285
    probaility of (+) in impasse = ~ 29%

- ODDS RATIO : exponentiated B1 COEFFICIENT
  - B1 = (slope of logit model = difference in log odds = log odds ratio
  - B1 = 0.901 is log odds ratio of (+) in impasse vs control
  - exp(b1) = exp(0.901) = 2.46
  - Ratio of odds in impasse are 2.46 times higher than in control. Bein in the impasse condition yields odds athat are 2.46 X higher than in control.

MARGINAL
total = 330 success : 71, failure : 259
p(+) = 71 / 330 = 0.215 = 22%
odds(+) = 71 / 259 = 0.274
CONTROL total = 158 success = 22; failure = 136
p(+) = 22/158 = 0.139 = 14%
odds(+) = 22/136 = 0.162
IMPASSE total = 172 success = 49; failure = 123
p(+) = 49/172 = 0.285 = 29%
odds(+) = 49/123 = 0.398

Visualize

```
print("MODEL PERFORMANCE")
```

```
[1] "MODEL PERFORMANCE"
```

```
performance(m1)
```

```
# Indices of model performance

AIC     |     BIC | Tjur's R2 |  RMSE | Sigma | Log_loss | Score_log | Score_spherical |   P
-------------------------------------------------------------------------------------------------
337.074 | 344.673 |     0.031 | 0.404 | 1.008 |    0.505 |   -16.847 |           0.021 | 0.6
```

```
print("SANITY CHECK REPORTING")
```

```
[1] "SANITY CHECK REPORTING"
```

```
report(m1)
```

```
Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

We fitted a logistic model (estimated using ML) to predict accuracy with pretty_condition (fo

  - The effect of pretty condition [impasse] is statistically significant and positive (beta

Standardized parameters were obtained by fitting the model on a standardized version of the d
```

```
print("MODEL PREDICTIONS")
```

```
[1] "MODEL PREDICTIONS"
```

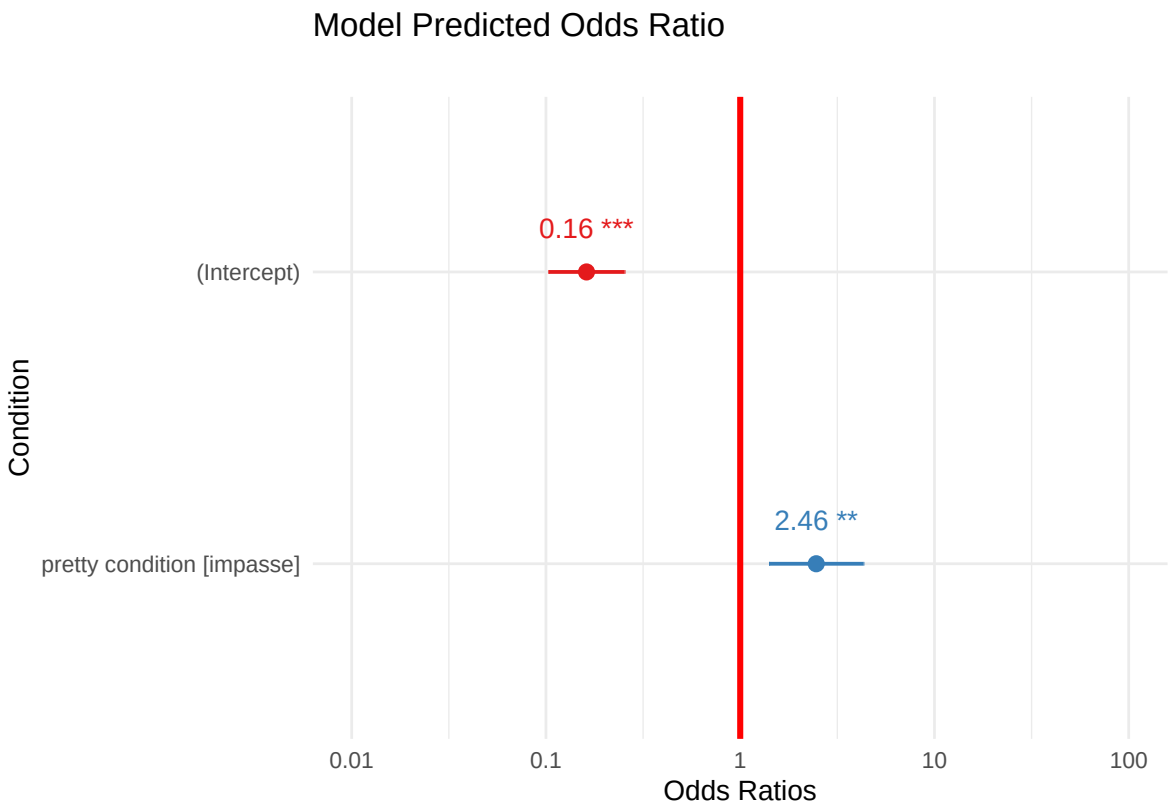
```
# Retrieve predictions as probabilities
# (for each level of the predictor)
p.control <- predict(m1,data.frame(pretty_condition="control"),type="response")
paste("Probability of success in control,", p.control)
```

```
[1] "Probability of success in control, 0.139240506329147"
```

```
p.impasse <- predict(m1,data.frame(pretty_condition="impasse"),type="response")
paste("Probability of success in impasse,", p.impasse)
```

```
[1] "Probability of success in impasse, 0.284883720930631"
```

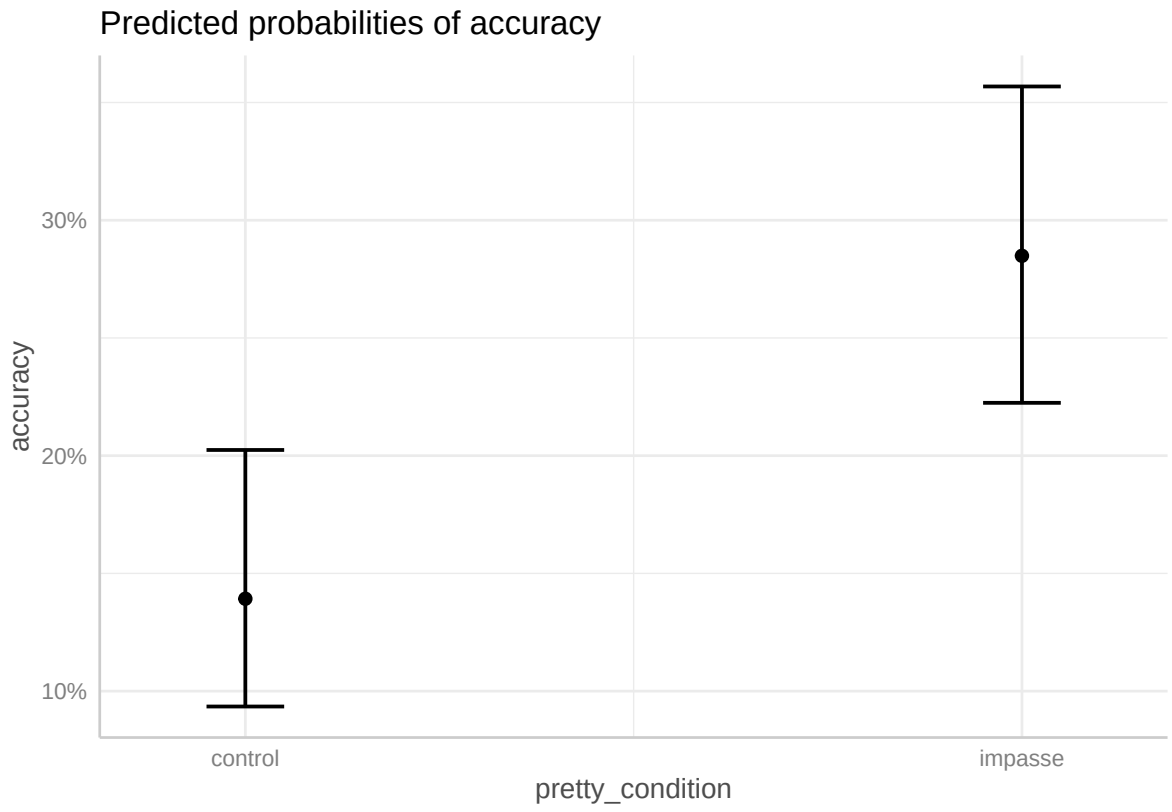
```
#: PLOT

#GGSTATS | MODEL | LOG ODDS
# library(ggstatsplot)
# ggcoefstats(m1, output = "plot") + labs(x = "Log Odds Estimate")

#SJPLOT | MODEL | ODDS RATIO
#library(sjPlot)
plot_model(m1, type="std2", vline.color = "red",
           show.intercept = TRUE,
           show.values = TRUE) +
  labs(title = "Model Predicted Odds Ratio",
       subtitle = "",
       x = "Condition")
```



```
#SJPLOT | MODEL | PROBABILITIES
# plot_model(m1, type="pred",
#            show.intercept = TRUE,
#            show.values = TRUE,
#            title = "Model Predicted Probability of Accuracy",
#            axis.title = c("Condition","Probability of Accurate Response"))

#GGEFFECTS | MODEL | PROBABILITIES
# library(ggeffects)
ggeffect(model = m1) %>% plot()
```

```
Package `effects` is not available, but needed for `ggeffect()`. Either install package `effe
```

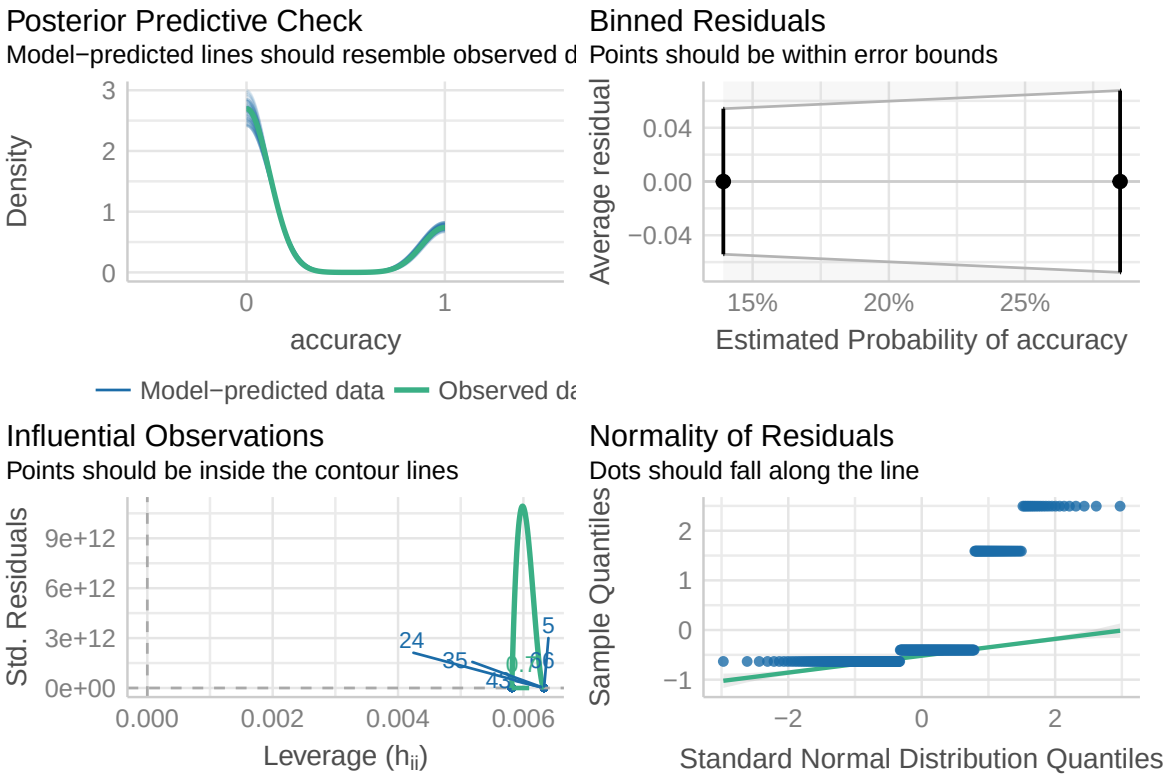
```
$pretty_condition
```



```
#SANITY CHECK SJPLOT
# library(effects)
# plot(allEffects(m))
```

Diagnostics

```
check_model(m1)
```




```
binned_residuals(m1)
```

```
Ok: About 100% of the residuals are inside the error bounds.
```

Inference

We fit a logistic regression model to analyze the effect of experimental condition on probability of a correct answer on the first question. In this model, the effect of condition is statistically significant (z = 3.16, p = 0.0016). The model predicts that the odds of a correct response on the first question in the *impasse condition* increase by 146% (e^{beta_1} = 2.46, 95% CI [1.42, 4.37]) over the *control condition*.

Equivalent statements:

- being in impasse condition increases log odds of correct response by 0.901 (over control)
- being in impasse increases odds of correct response in impasse over control increases by factor of 2.46
- probability of correct response in control predicted as 28.5%, vs only 14% in control condition

```
#PRETTY TABLE SJPLOT
tab_model(m1)
```

		accuracy	
Predictors	Odds Ratios	CI	p
(Intercept)	0.16	0.10 – 0.25	**<0.001**
pretty condition [impasse]	2.46	1.42 – 4.37	**0.002**
Observations	330		
R^2 Tjur	0.031		

TODO ORDINAL REGRESSION

Fit an ordinal logistic regression (at the subject level), predicting Q1 interpretation by condition.

- https://stats.oarc.ucla.edu/r/faq/ologit-coefficients/
- https://journals.sagepub.com/doi/full/10.1177/2515245918823199
- todo see ordinal regression video

```
# #CREATE DATAFRAME OF Q1
# df <- df_items %>% filter(q ==1) %>% mutate(scaled = as.factor(score_SCALED))
#
# #MODEL
# m <- polr(scaled ~ condition , data = df, Hess=TRUE)
```

```
# summary(m)
# confint(m)
# performance(m)
# report(m)
#
# #exponentiate coefficients and CIs
# ci <- confint(m)
# ci
# e <- coef(m)
# e
# # exp(cbind(e,ci))
#
# # Retrieve predictions as probabilities
# # (for each level of the predictor)
# # p.control <- predict(m,data.frame(condition="111"),type="response")
# # paste("Probability of success in control,", p.control)
# # p.impasse <- predict(m,data.frame(condition="121"),type="response")
# # paste("Probability of success in impasse,", p.impasse)
#
# # Plot Predicted data and original data points
# # ggplot(df, aes(x=condition, y=accuracy)) +
# #   geom_point() +
# #   stat_smooth(method="glm", color="green", se=FALSE,
# #                 method.args = list(family=binomial))
#
# #TO PLOT ALL EFFECTS
# library(effects)
# plot(allEffects(m))
#
# #SJPLOT
# library(sjPlot)
# plot_model(m, )
#
#
# #CONVERT TO PROBABILITIES
# newdat <- data.frame(condition=c("111","121"))
# prob <- (phat <- predict(object = m, newdat, type="p"))
# prob
#
```

REPEATED ITEM LEVEL

Test Phase Accuracy (absolute score)

Mixed Logistic Regression

Fit a mixed logistic regression (at the item level), predicting accuracy (absolute score) on test phase questions by condition; accounting for random effects of subject.

0.0.0.0.1 * Fit Model

```
#SETUP DATA
#PREPARE DATA
n_items = 8 #number of items in test

#item level
df_test = df_items %>% filter(q %nin% c(1,2,3,4,5,6,9)) %>% mutate(
  accuracy = as.factor(score_niceABS),
  q = as.factor(q)
)

df <- df_test

library(lmerTest) #for CIs in glmer

## 1 | SETUP RANDOM EFFECT

#:: EMPTY MODEL (baseline, no random effect)
m0 = glm(accuracy ~ 1, family = "binomial", data = df)

#:: RANDOM INTERCEPT SUBJECT
mm.rS <- glmer(accuracy ~ (1|subject), data = df,family = "binomial")

# :: TEST random effect
paste("AIC with random effect is lower than glm empty model?", m0$aic > AIC(logLik(mm.rS))
```

```
[1] "AIC with random effect is lower than glm empty model? TRUE"
```

```
test_lrt(m0,mm.rS) #same as anova(m0, m1, test = "Chi")
```

```
# Likelihood-Ratio-Test (LRT) for Model Comparison (ML-estimator)

Name  |    Model | df | df_diff |    Chi2 |      p
-------------------------------------------------
m0    |      glm |  1 |         |         |
mm.rS | glmerMod |  2 |       1 | 1783.73 | < .001
```

```
paste("Likelihood Ratio test is significant? p = ",(test_lrt(m0,mm.rS))$p[2])
```

```
[1] "Likelihood Ratio test is significant? p =  0"
```

```
## 2 | ADD FIXED EFFECT

# SUBJECT INTERCEPT | FIXED CONDITION
mm.CrS <- glmer(accuracy ~ pretty_condition + (1|subject),
                data = df,family = "binomial")

# :: TEST fixed factor
paste("AIC with random effect is lower than glm empty model?", AIC(logLik(mm.rS)) > AIC(lo
```

```
[1] "AIC with random effect is lower than glm empty model? TRUE"
```

```
test_lrt(mm.rS,mm.CrS) #same as anova(m0, m1, test = "Chi")
```

```
# Likelihood-Ratio-Test (LRT) for Model Comparison (ML-estimator)

Name   |    Model | df | df_diff | Chi2 |     p
-----------------------------------------------
mm.rS  | glmerMod |  2 |         |      |
mm.CrS | glmerMod |  3 |       1 | 4.98 | 0.026
```

```
paste("Likelihood Ratio test is significant? p = ",(test_lrt(mm.rS,mm.CrS))$p[2])
```

```
[1] "Likelihood Ratio test is significant? p =  0.0256331468201315"
```

0.0.0.0.2 * Visualize

```r
#: PRINT MODEL
print("PREDICTOR MODEL")
```

```
[1] "PREDICTOR MODEL"
```

```r
summary(mm.CrS)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: accuracy ~ pretty_condition + (1 | subject)
   Data: df

     AIC      BIC   logLik deviance df.resid
    1385     1402     -689     1379     2637

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.5307 -0.0193 -0.0097  0.1135  2.7426

Random effects:
 Groups  Name        Variance Std.Dev.
 subject (Intercept) 117      10.8
Number of obs: 2640, groups:  subject, 330

Fixed effects:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)               -9.182      0.677  -13.56   <2e-16 ***
pretty_conditionimpasse    1.632      0.753    2.17     0.03 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
prtty_cndtn -0.394
```

```r
#: INTERPRET COEFFICIENTS

print("MODEL PERFORMANCE")
```

```
[1] "MODEL PERFORMANCE"
```

```
performance(mm.CrS)
```

```
# Indices of model performance

AIC      |     AICc |      BIC | R2 (cond.) | R2 (marg.) |   ICC |  RMSE | Sigma | Log_loss
-----------------------------------------------------------------------------------------
1384.749 | 1384.758 | 1402.385 |      0.973 |      0.005 | 0.973 | 0.203 | 1.000 |    0.130
```

```
print("SANITY CHECK REPORTING")
```

```
[1] "SANITY CHECK REPORTING"
```

```
report(mm.CrS)
```

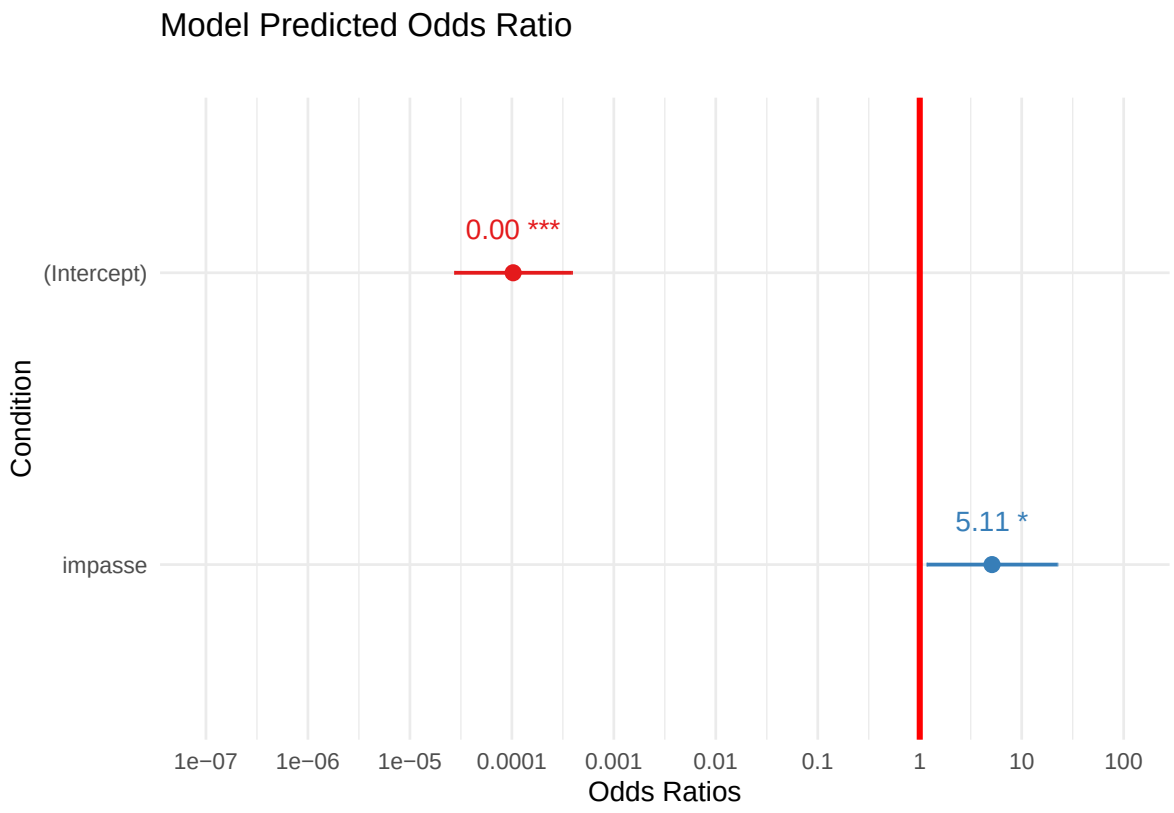
```
Package 'merDeriv' needs to be installed to compute confidence intervals
  for random effect parameters.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Package 'merDeriv' needs to be installed to compute confidence intervals
  for random effect parameters.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.
```

We fitted a logistic mixed model (estimated using ML and Nelder-Mead optimizer) to predict ac

- The effect of pretty condition [impasse] is statistically significant and positive (beta

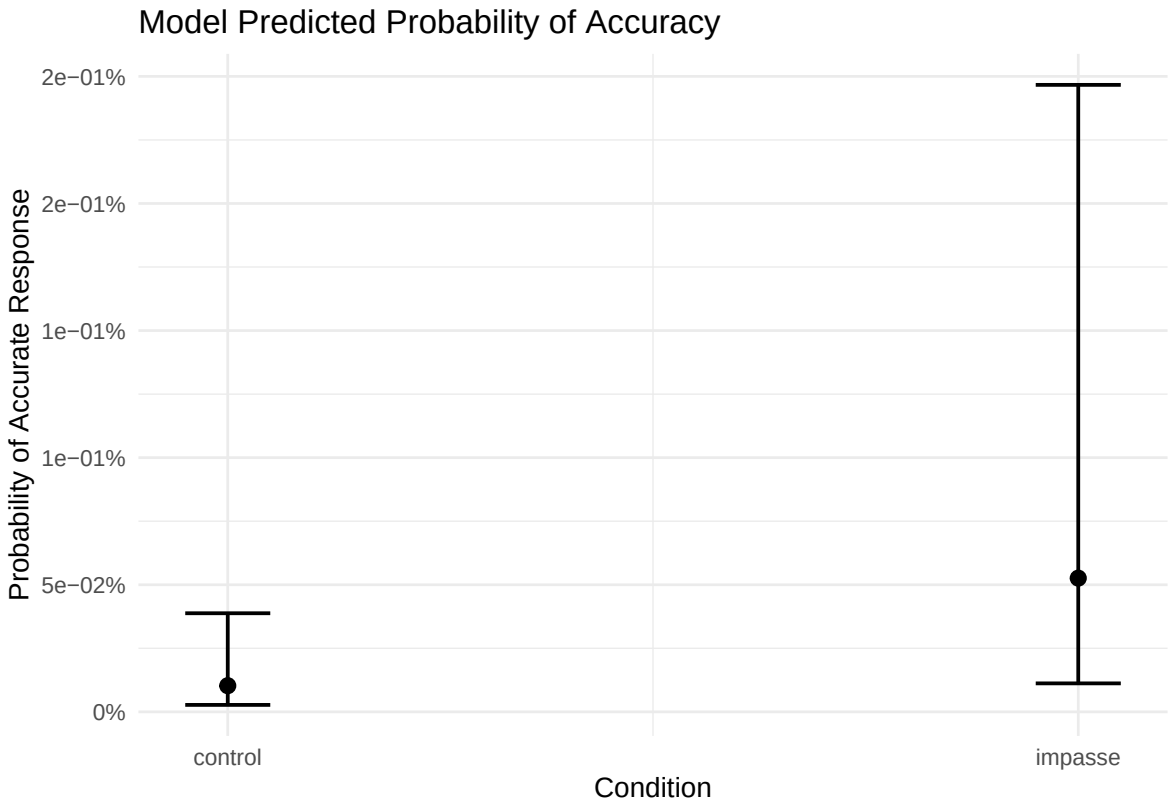
Standardized parameters were obtained by fitting the model on a standardized version of the c

```
#: PLOT

#SJPLOT | MODEL | ODDS RATIO
#library(sjPlot)
plot_model(mm.CrS, type="std2", vline.color = "red",
           show.intercept = TRUE,
           show.values = TRUE) +
  labs(title = "Model Predicted Odds Ratio",
       subtitle = "",
       x = "Condition")
```



```
#SJPLOT | MODEL | PROBABILITIES
plot_model(mm.CrS, type="pred",
           show.intercept = TRUE,
           show.values = TRUE,
           title = "Model Predicted Probability of Accuracy",
           axis.title = c("Condition","Probability of Accurate Response"))
```

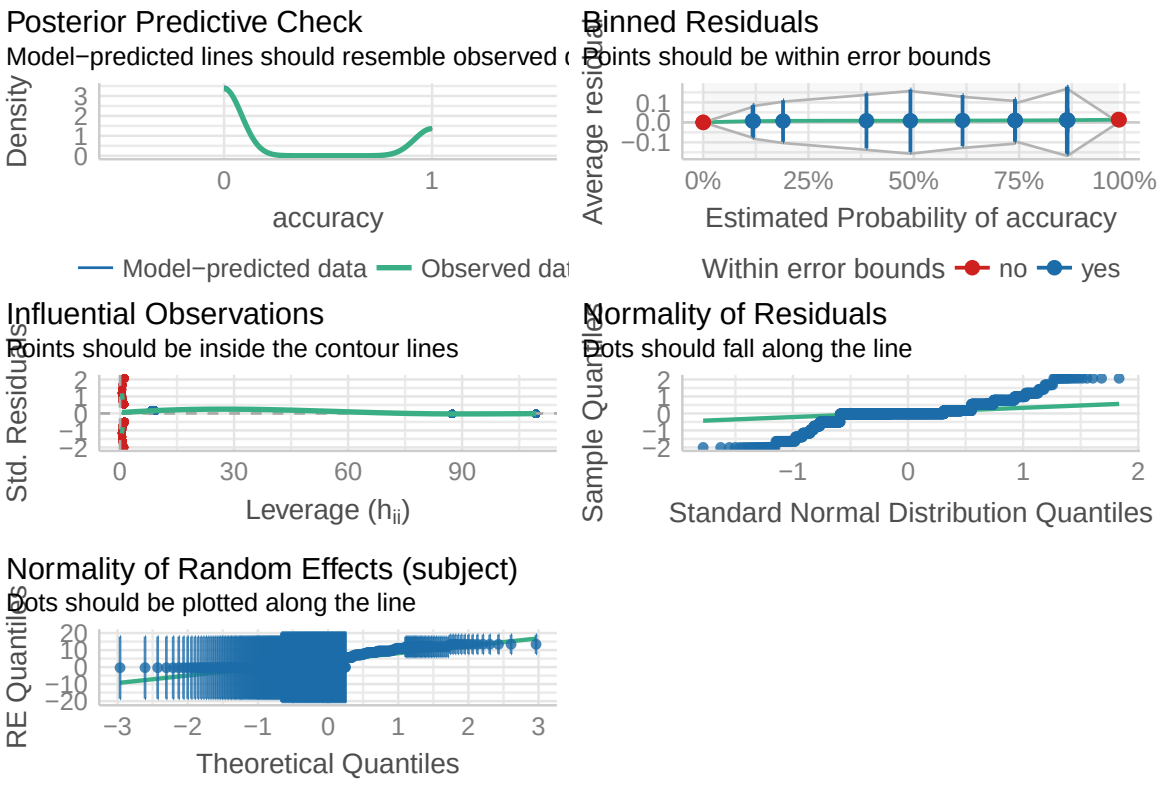
```
$pretty_condition
```




```
#GGEFFECTS | MODEL | PROBABILITIES
# library(ggeffects)
# ggeffect(model = mm.CrS) %>% plot()

#SANITY CHECK SJPLOT
# library(effects)
# plot(allEffects(mm.CrS))
```

0.0.0.0.3 * Diagnostics

```
check_model(mm.CrS)
```




```
binned_residuals(mm.CrS)
```

```
Warning: Probably bad model fit. Only about 75% of the residuals are inside the error bounds
```

0.0.0.0.4 * Inference

We fit a mixed-effect binomial logistic regression model with random intercepts for subjects to investigate the effect of condition on test phase item accuracy. The model including a fixed effect of condition performed significantly better than an intercept-only baseline model (2(3): 4.98, $p < 0.05$). Consistent with the pattern of results for the first question only, across all test-phase items, being in the impasse condition increases the odds of a correct response by a factor of 5 over the control condition $e^{\beta_1} = 5.11$, 95% CI [1.17,22,36], $p < 0.05$.

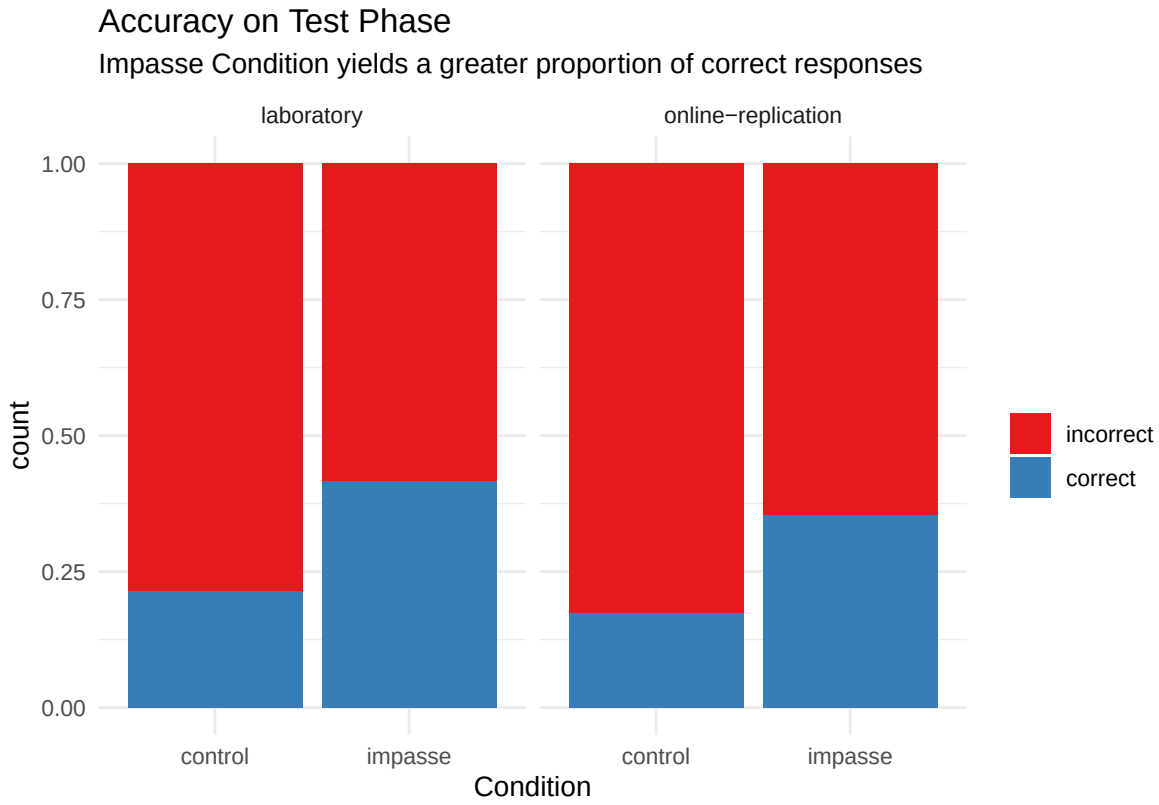
```
# PRETTY TABLE SJPLOT
tab_model(mm.CrS)
```

TODO Mixed Ordinal Regression

SUBJECT-LEVEL

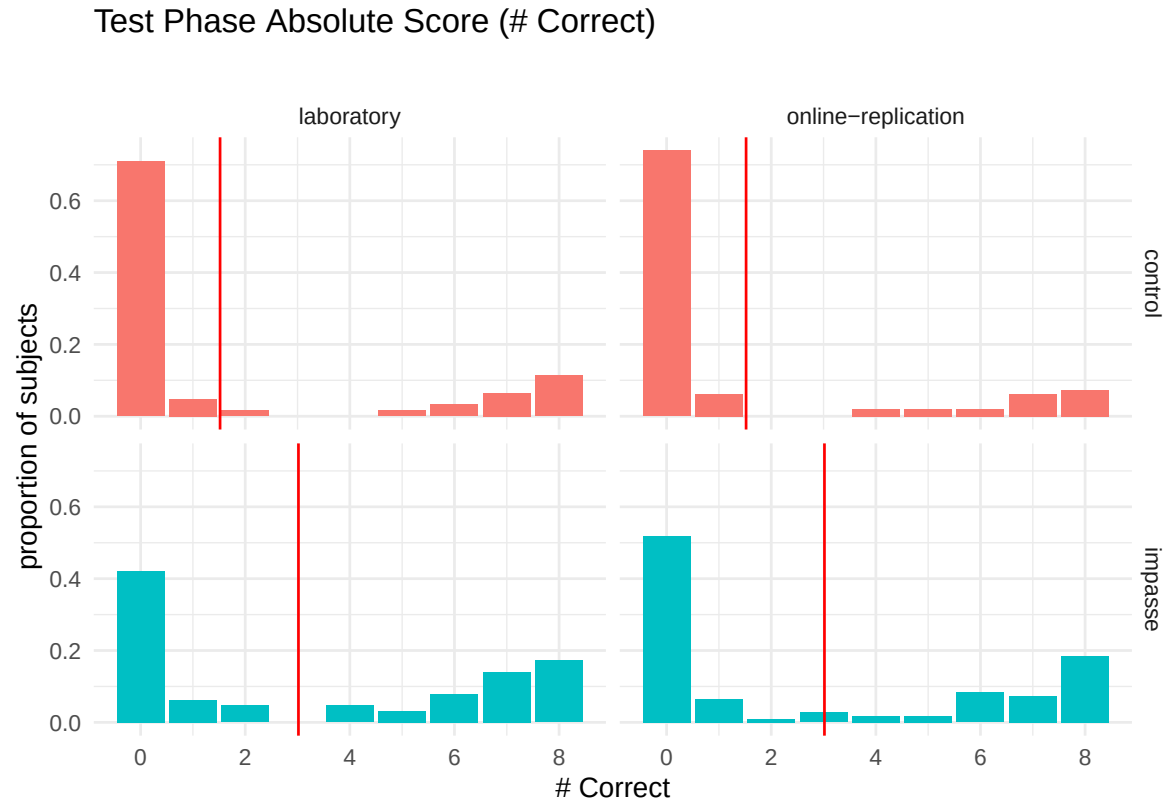
Test Phase Absolute Score (# questions)

```
#PREPARE DATA
n_items = 8 #number of items in test

#item level
df = df_items %>% filter(q %nin% c(1,2,3,4,5,6,9)) %>% mutate(
  accuracy = recode_factor(score_niceABS, "0" ="incorrect","1"="correct"),
  q = as.factor(q)
)

#STACKED PROPORTIONAL BAR CHART
df %>%
  ggplot(data = .,
         mapping = aes(x = pretty_condition,
                       fill = accuracy)) +
  geom_bar(position = "fill" ) + #,color = "black") +
  scale_fill_brewer(palette = "Set1")  +
  facet_wrap(~pretty_mode) +
   labs(#y = "",
        title = "Accuracy on Test Phase",
        x = "Condition",
        fill = "",
        subtitle="Impasse Condition yields a greater proportion of correct responses")
```




```
#GROUPED PROPORTIONAL BAR CHART
# gf_props(~accuracy, fill = ~pretty_condition, x =~pretty_condition,
#        position = position_dodge(), data = df) %>%
#   gf_facet_grid(~pretty_mode) +
#    labs(x = "Correct Responses in Test Phase",
#        title = "Accuracy on Task by Condition",
#        subtitle="Impasse Condition yields a greater proportion of correct responses")


#FACETED HISTOGRAM
stats = df_subjects %>% group_by(pretty_condition) %>% dplyr::summarise(mean = mean(item_t
gf_props(~item_test_NABS,
         fill = ~pretty_condition, data = df_subjects) %>%
  gf_facet_grid(pretty_condition ~ pretty_mode) %>%
  gf_vline(data = stats, xintercept = ~mean, color = "red") +
  labs(x = "# Correct",
       y = "proportion of subjects",
```

```
        title = "Test Phase Absolute Score (# Correct)",
        subtitle = "") + theme(legend.position = "blank")
```



Linear Regression

LM on Test Phase absolute score ***as number of questions****, rather than % correct.*

```
#SCORE predicted by CONDITION
lm.1 <- lm(item_test_NABS ~ pretty_condition, data = df_subjects)
paste("Model")
```

```
[1] "Model"
```

```
summary(lm.1)
```

```
Call:
lm(formula = item_test_NABS ~ pretty_condition, data = df_subjects)

Residuals:
   Min     1Q Median     3Q    Max
 -3.02  -2.77  -1.52   2.98   6.48

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                1.519      0.251    6.04  4.1e-09 ***
pretty_conditionimpasse    1.498      0.348    4.30  2.2e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.16 on 328 degrees of freedom
Multiple R-squared:  0.0535,    Adjusted R-squared:  0.0506
F-statistic: 18.5 on 1 and 328 DF,  p-value: 0.0000222
```

```
paste("Partition Variance")
```

```
[1] "Partition Variance"
```

```
anova(lm.1)
```

```
Analysis of Variance Table

Response: item_test_NABS
                  Df Sum Sq Mean Sq F value   Pr(>F)
pretty_condition    1    185     185    18.5 0.000022 ***
Residuals         328   3274      10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
paste("Confidence Interval on Parameter Estimates")
```

```
[1] "Confidence Interval on Parameter Estimates"
```

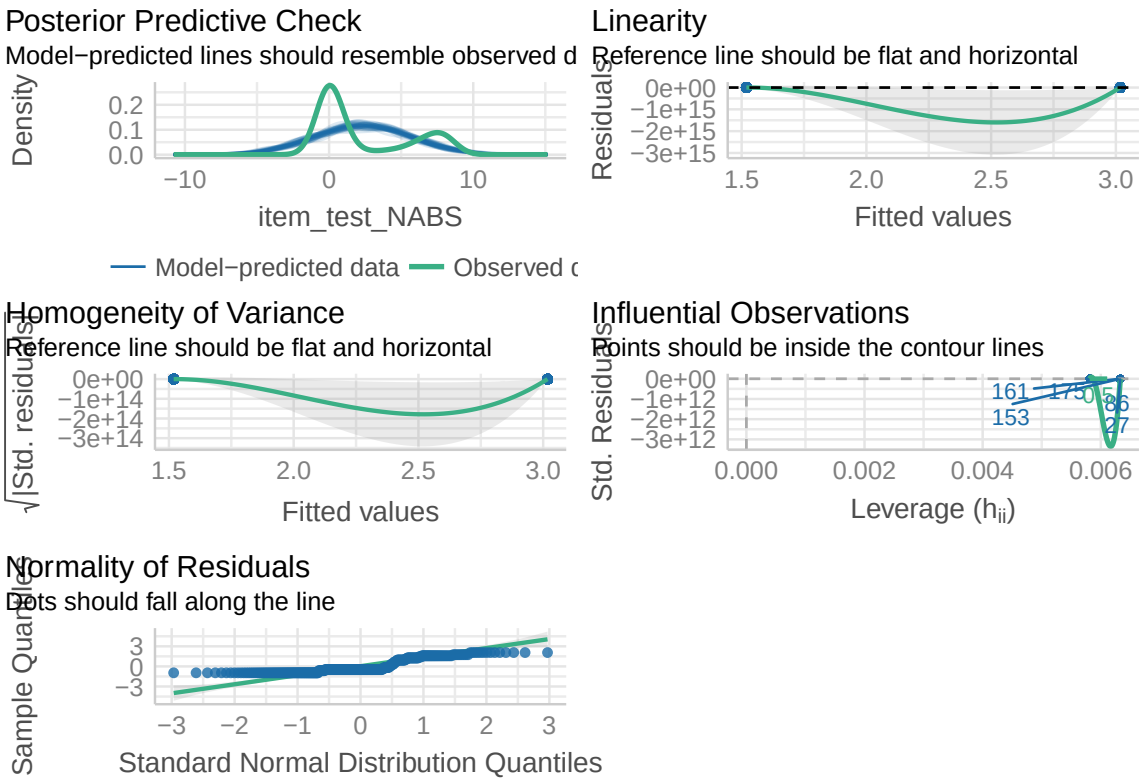
```r
confint(lm.1)
```

```
                       2.5 % 97.5 %
(Intercept)            1.025   2.01
pretty_conditionimpasse 0.814   2.18
```

```r
report(lm.1) #sanity check
```

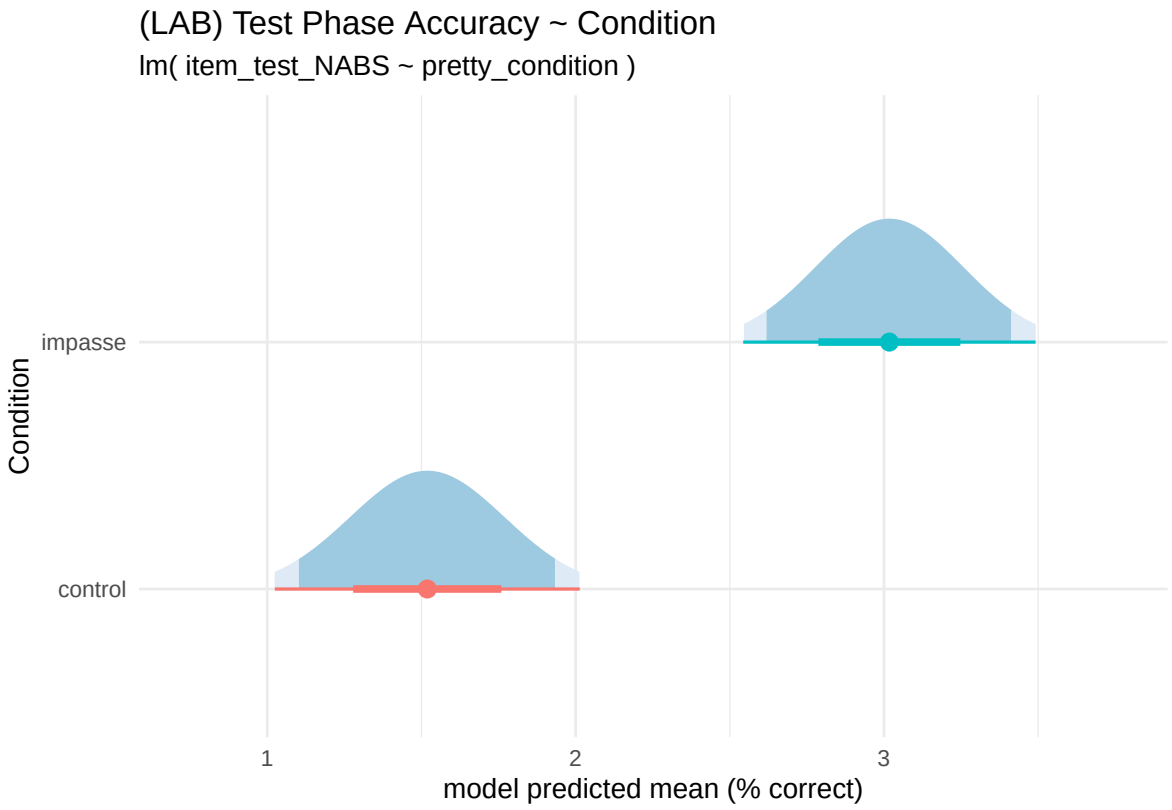
```
Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

We fitted a linear model (estimated using OLS) to predict item_test_NABS with pretty_conditio

  - The effect of pretty condition [impasse] is statistically significant and positive (beta

Standardized parameters were obtained by fitting the model on a standardized version of the d
```

```r
check_model(lm.1)
```




```
#MODEL ESTIMATES WITH UNCERTAINTY

#setup references
m <- lm.1
df <- df_subjects
call <- m$call %>% as.character()

# uncertainty model visualization
df  %>%
  modelr::data_grid(pretty_condition) %>%
  augment(lm.1, newdata = ., se_fit = TRUE) %>%
  ggplot(aes(y = pretty_condition, color = pretty_condition)) +
  stat_halfeye( scale = .5,
      aes(
        xdist = dist_student_t(df = df.residual(m), mu = .fitted, sigma = .se.fit),
        fill = stat(cut_cdf_qi(cdf,
                .width = c(.90, .95),
```

```
                labels = scales::percent_format()))))) +
scale_fill_brewer(direction = -1) +
labs (title = "(LAB) Test Phase Accuracy ~ Condition",
      x = "model predicted mean (% correct)", y = "Condition", fill = "Interval",
      subtitle = paste("lm(",call[2],")")
) + theme(legend.position = "blank")
```



Poisson Regression

https://stats.oarc.ucla.edu/r/dae/poisson-regression/

The outcome variable absolute score is clearly not normal. As it represents the cumulative number of items a participant has answered correctly, we can consider it a type of *count*, (ie. count of the number of questions the participant got correct) and attempt to model it using a General Linear Model with the Poisson distribution (and the default log-link function).

```r
#POISSON

#SCORE predicted by CONDITION --> POISSON DISTRIBUTION
p.1 <- glm(item_test_NABS ~ pretty_condition, data = df_subjects, family = "poisson")
paste("Model")
```

```
[1] "Model"
```

```r
summary(p.1)
```

```
Call:
glm(formula = item_test_NABS ~ pretty_condition, family = "poisson",
    data = df_subjects)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
 -2.46   -2.28   -1.74    1.51    3.69

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)               0.4180     0.0645    6.48  9.4e-11 ***
pretty_conditionimpasse   0.6864     0.0781    8.79  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1579.3  on 329  degrees of freedom
Residual deviance: 1496.7  on 328  degrees of freedom
AIC: 1956

Number of Fisher Scoring iterations: 6
```

```r
paste("Partition Variance")
```

```
[1] "Partition Variance"
```

```r
anova(p.1)
```

```
Analysis of Deviance Table

Model: poisson, link: log

Response: item_test_NABS

Terms added sequentially (first to last)

                 Df Deviance Resid. Df Resid. Dev
NULL                                329       1579
pretty_condition  1     82.7       328       1497
```

```
paste("Confidence Interval on Parameter Estimates")
```

```
[1] "Confidence Interval on Parameter Estimates"
```

```
confint(p.1)
```

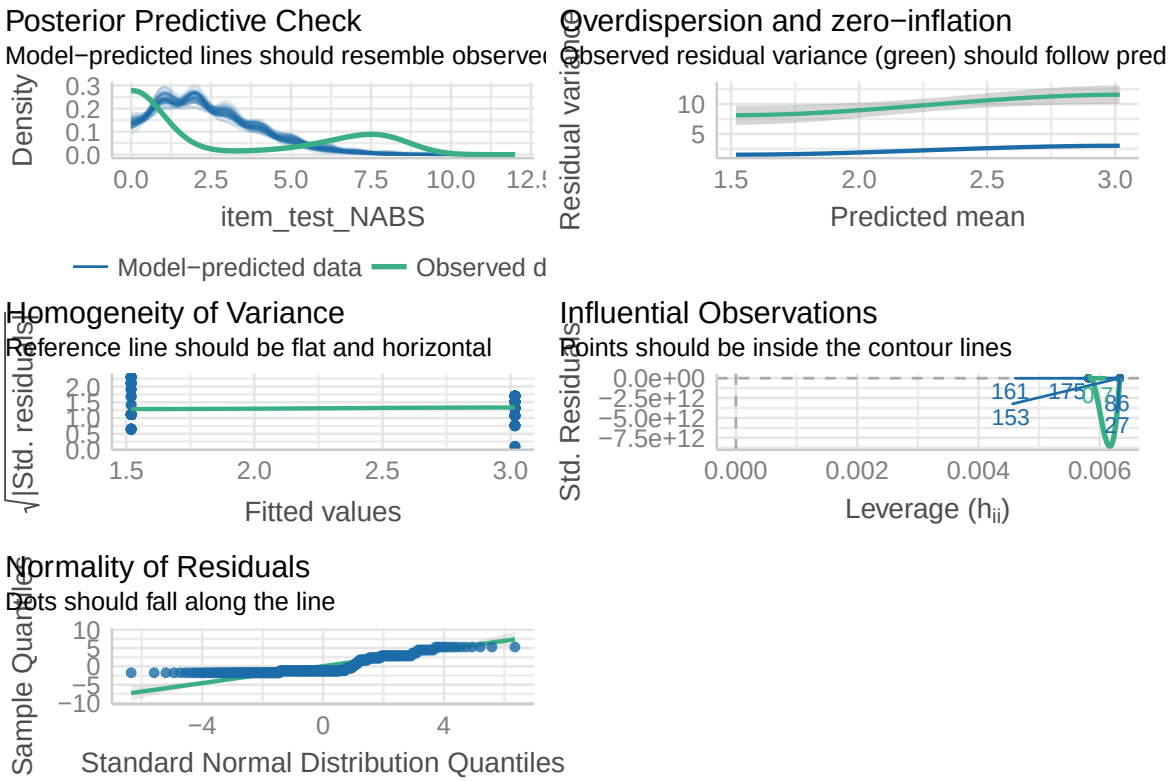
```
Waiting for profiling to be done...

                        2.5 % 97.5 %
(Intercept)             0.289  0.542
pretty_conditionimpasse 0.535  0.841
```

```
report(p.1) #sanity check
```

```
Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.
```

```
We fitted a poisson model (estimated using ML) to predict item_test_NABS with pretty_conditio

  - The effect of pretty condition [impasse] is statistically significant and positive (beta

Standardized parameters were obtained by fitting the model on a standardized version of the o
```

```
check_model(p.1)
```



Zero Inflated Poisson

https://stats.oarc.ucla.edu/r/dae/zip/
Poisson count process with excess zeros

```
#ZERO INFLATED POISSON

zinfp.1 <- zeroinfl(item_test_NABS ~  item_q1_rt| pretty_condition , data = df_subjects)
summary(zinfp.1)
```

```
Call:
zeroinfl(formula = item_test_NABS ~ item_q1_rt | pretty_condition, data = df_subjects)

Pearson residuals:
   Min     1Q Median     3Q    Max
-0.934 -0.821 -0.548  0.965  2.421

Count model coefficients (poisson with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 1.654243   0.059975   27.58   <2e-16 ***
item_q1_rt  0.001690   0.000849    1.99    0.047 *

Zero-inflation model coefficients (binomial with logit link):
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                0.978      0.179    5.46  4.7e-08 ***
pretty_conditionimpasse   -1.055      0.236   -4.48  7.5e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 7
Log-likelihood: -531 on 4 Df
```

```
report(zinfp.1)
```

```
Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

We fitted a zero-inflated poisson model to predict item_test_NABS with item_q1_rt and pretty_
```

```
  - The effect of item q1 rt is statistically significant and positive (beta = 1.69e-03, 95%
  - The effect of pretty condition [impasse] is statistically significant and negative (beta

Standardized parameters were obtained by fitting the model on a standardized version of the
```

```
performance(zinfp.1)
```

```
# Indices of model performance

AIC      |      BIC |    R2 | R2 (adj.) |  RMSE | Sigma | Score_log | Score_spherical
-------------------------------------------------------------------------------------------
1070.173 | 1085.370 | 0.354 |     0.350 | 3.131 | 3.150 |    -1.609 |           0.044
```

```
# check_model(zinfp.1)
```

Negative Binomial Regression

https://stats.oarc.ucla.edu/r/dae/negative-binomial-regression/ - overdispersed count data (variance much greater than mean)

```
#NEGATIVE BIONOMIAL REGRESSION
# - https://stats.oarc.ucla.edu/r/dae/negative-binomial-regression/
# - Overdispersed Count variables

library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select
```

```
nb.1 <- glm.nb(item_test_NABS ~ pretty_condition, data = df_subjects)
summary(nb.1)
```

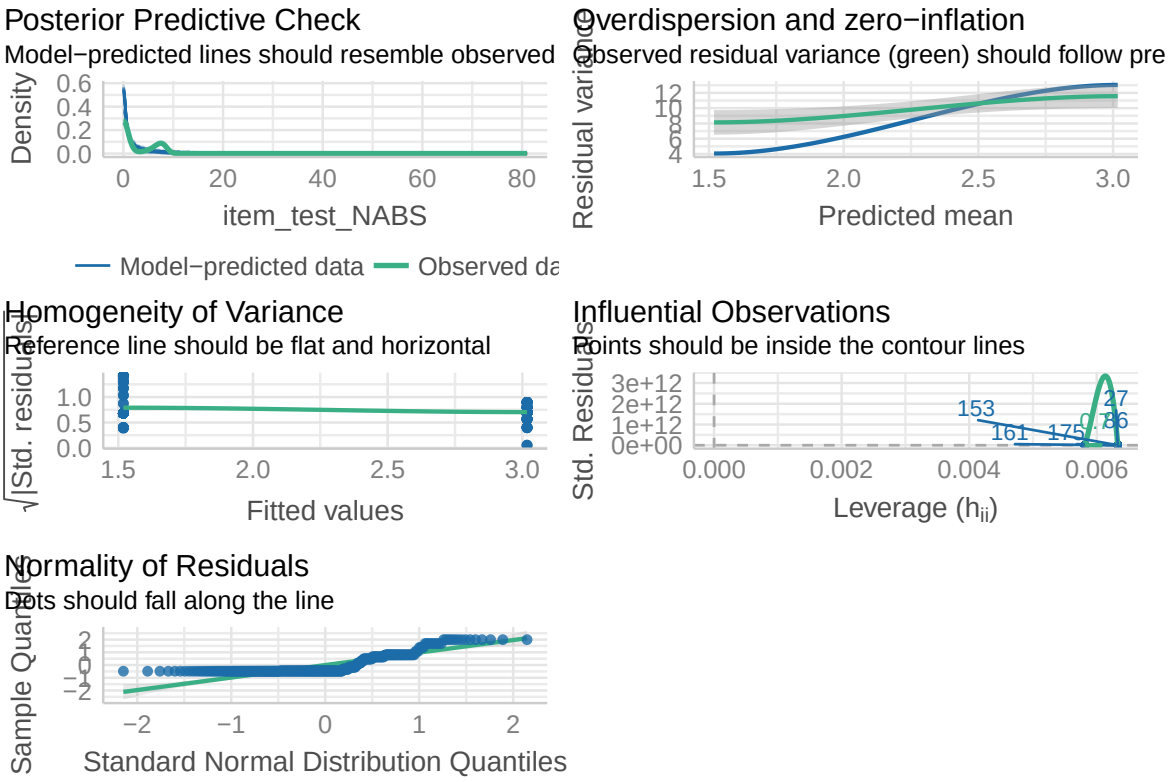
```
Call:
glm.nb(formula = item_test_NABS ~ pretty_condition, data = df_subjects,
    init.theta = 0.253501538, link = log)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.139  -1.102  -0.993   0.378   1.091

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                0.418      0.171    2.45   0.0143 *
pretty_conditionimpasse    0.686      0.232    2.95   0.0031 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(0.254) family taken to be 1)

    Null deviance: 279.52  on 329  degrees of freedom
Residual deviance: 270.97  on 328  degrees of freedom
AIC: 1194

Number of Fisher Scoring iterations: 1

              Theta:  0.2535
          Std. Err.:  0.0315

 2 x log-likelihood:  -1188.1290
```

```
report(nb.1)
```

```
Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.
```

```
Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

We fitted a negative-binomial model (estimated using ML) to predict item_test_NABS with pret

  - The effect of pretty condition [impasse] is statistically significant and positive (beta

Standardized parameters were obtained by fitting the model on a standardized version of the
```

```r
check_model(nb.1)
```




```r
#check model assumption
#assumes conditional means are not equal to conditional variances
#conduct likelihood ration test to compare and test [need poisson]
m3 <- glm(item_test_NABS ~ pretty_condition, family = "poisson", data = df_subjects)
pchisq(2 * (logLik(nb.1) - logLik(m3)), df = 1, lower.tail = FALSE)
```

```
'log Lik.' 4.3e-168 (df=3)
```

```
#A large (+) log likelihood suggests that the negative binomial is more appropriate than t
```

```
#EXPONENTIATE PARAMETER ESTIMATES
est <- cbind(Estimate = coef(nb.1), confint(nb.1))
```

```
Waiting for profiling to be done...
```

```
#exponentiate parameter estimates
print("Exponentiated Estimates")
```

```
[1] "Exponentiated Estimates"
```

```
exp(est)
```

```
                        Estimate 2.5 % 97.5 %
(Intercept)                 1.52  1.10   2.15
pretty_conditionimpasse     1.99  1.26   3.13
```

The variable condition has a coefficient of 0.67, ($p < 0.005$). This means that for the impasse condition, the expected log count # of questions increases by 0.67. By exponentiating the estimate we see that # question correct rate for the impasse condition is nearly 2x that of the control condition.

Diagnostics ??

Zero Inflated Negative Binomial Regression

https://stats.oarc.ucla.edu/r/dae/zinb/ count data that are overdispersed and have excess zeros

Zero-inflated negative binomial regression is for modelling count variables with excessive zeros, and especially when the count data are overdispersed (mean is much larger than variance). It can help account for situations where theory suggests that excess zeros are generated by 2 separate processes, one that includes the other count values, and the other that is just the zeros, and thus that the *excess* zeros can be modelled independently.

Total Absolute Score (# items correct) may fit this situation, as the data are overdispersed (variance much greater than the mean) and there are are very large number of zeros. It is theoretically plausible that these excess zeros (no answers correct) are the result of a different 'process' … (i.e) little understanding and/or resistance to restructuring understanding of the coordinate system. However, I am not certain if it is plausible to suggest that the zeros themselves are the result of two different processes: (ie. perhaps trying to understand, and not trying to understand?) <- this could maybe be disentangled by first question latency?

The model includes: - A logistic model to model which of the two processes the zero outcome is associated with - A negative binomial model to model the count process

```
library(pscl) #  for zeroinfl negbinomial

#ZERO INFLATED NEGATIVE BINOMIAL
zinb.1 <- zeroinfl(item_test_NABS ~ pretty_condition | pretty_condition , data = df_subjec
#before the | is the count part, after the | is the logit model
paste("Model")
```

```
[1] "Model"
```

```
summary(zinb.1)
```

```
Call:
zeroinfl(formula = item_test_NABS ~ pretty_condition | pretty_condition,
    data = df_subjects, dist = "negbin")

Pearson residuals:
   Min     1Q Median     3Q    Max
-0.866 -0.794 -0.538  0.856  2.294

Count model coefficients (negbin with log link):
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)              1.7126     0.0728   23.54  < 2e-16 ***
pretty_conditionimpasse  0.0451     0.0880    0.51  0.60810
Log(theta)               3.1851     0.8732    3.65  0.00026 ***

Zero-inflation model coefficients (binomial with logit link):
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)               0.974      0.179    5.43  5.5e-08 ***
pretty_conditionimpasse  -1.056      0.236   -4.47  7.7e-06 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta = 24.169
Number of iterations in BFGS optimization: 7
Log-likelihood: -532 on 5 Df
```

```
report(zinb.1)
```

```
Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

Warning: 'data_findcols()' is deprecated and will be removed in a future update.
  Its usage is discouraged. Please use 'data_find()' instead.

We fitted a zero-inflated negative-binomial model to predict item_test_NABS with pretty_condi

  - The effect of pretty condition [impasse] is statistically non-significant and positive (
  - The effect of pretty condition [impasse] is statistically significant and negative (beta

Standardized parameters were obtained by fitting the model on a standardized version of the c
```

```
performance(zinb.1)
```

```
# Indices of model performance

AIC      |      BIC |    R2 | R2 (adj.) |  RMSE | Sigma | Score_log | Score_spherical
-----------------------------------------------------------------------------------------
1073.880 | 1092.876 | 0.363 |     0.359 | 3.150 | 3.174 |    -1.649 |           0.043
```

```
#   rootogram(zinb.1)



# #EXPONENTIATE PARAMETER ESTIMATES
# est <- cbind(Estimate = coef(zinb.1), confint(zinb.1))
# #exponentiate parameter estimates
# print("Exponentiated Estimates")
# exp(est)
```

In the count model, the coefficient for the condition is very small, and not significant (suggesting it does not contribute to the count yielding process?).

In the zero-inflation model, the coefficient for the condition variable is -1.056 and statistically significant. This suggests that the log odds of being an excessive zero decrease by 1.06 if you are in the impasse condition (exponentiate it?)

TODO come back to this and discuss further

Tobit Regression

https://stats.oarc.ucla.edu/r/dae/tobit-models/

For censored data (i.e. truncated axis). The tobit model, also called a censored regression model, is designed to estimate linear relationships between variables when there is either left- or right-censoring in the dependent variable (also known as censoring from below and above, respectively). Censoring from above takes place when cases with a value at or above some threshold, all take on the value of that threshold, so that the true value might be equal to the threshold, but it might also be higher. In the case of censoring from below, values those that fall at or below some threshold are censored.

```
#set up data
df <- df_subjects %>% mutate(
  accuracy = s_NABS
)

library(VGAM)
```

```
Loading required package: stats4
```

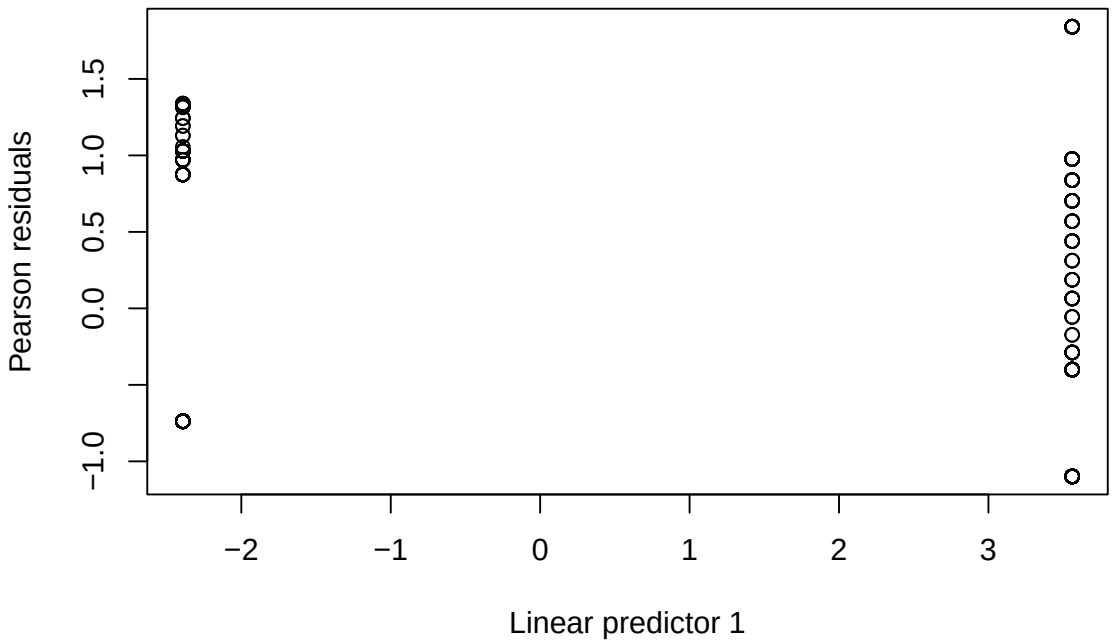
```
Loading required package: splines
```

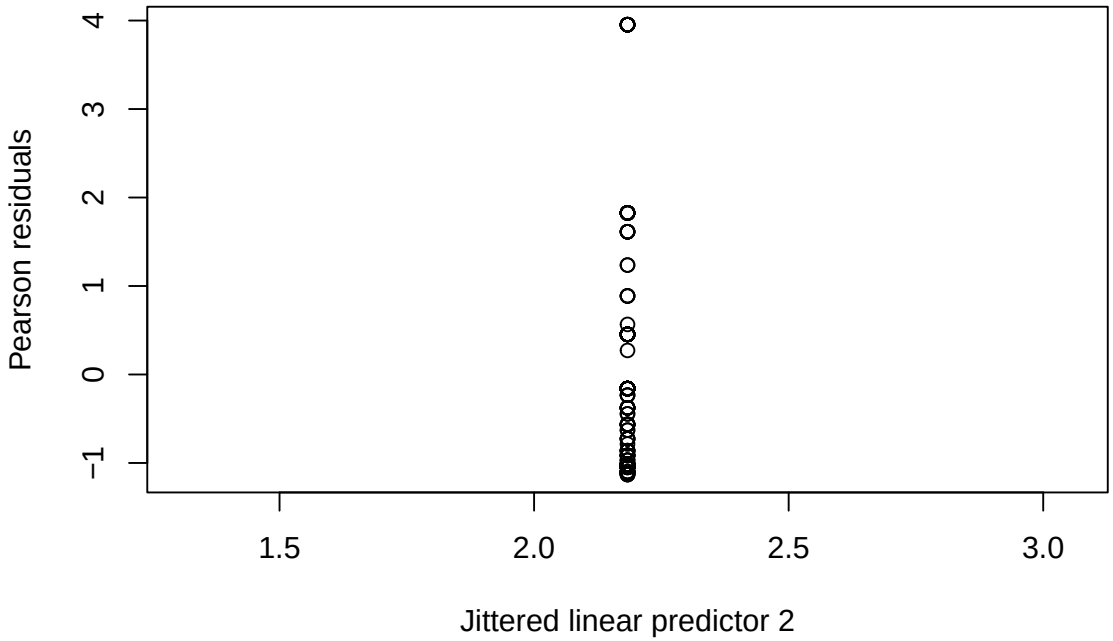
```
Attaching package: 'VGAM'

The following object is masked from 'package:distributional':

    cdf
```

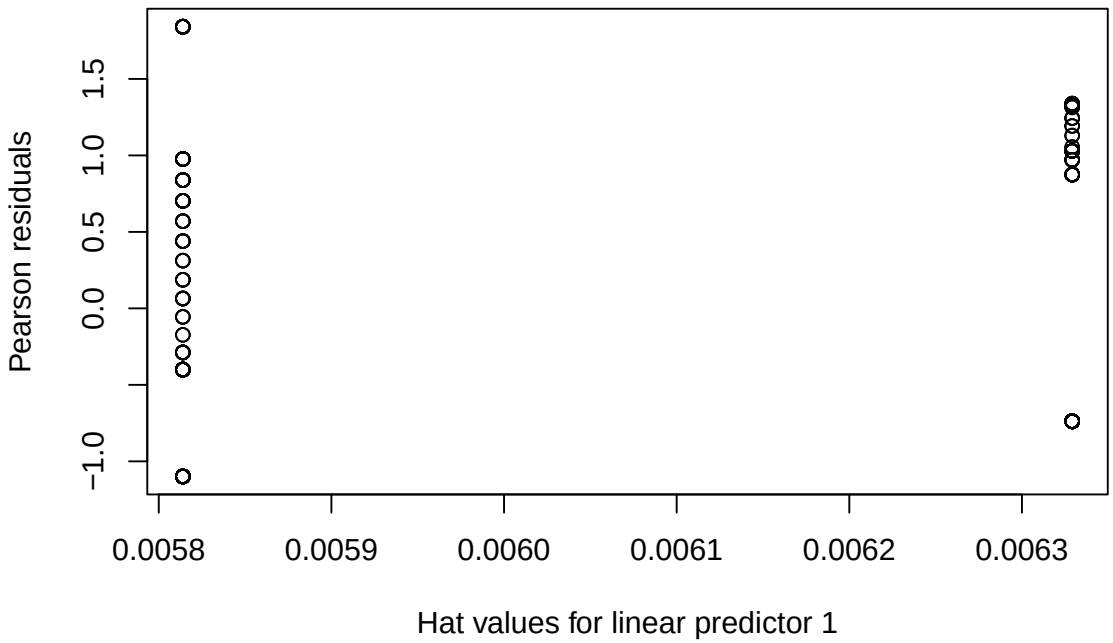
```r
t <- vglm(accuracy ~ condition, tobit(Upper = 13), data = df)
summary(t)
```

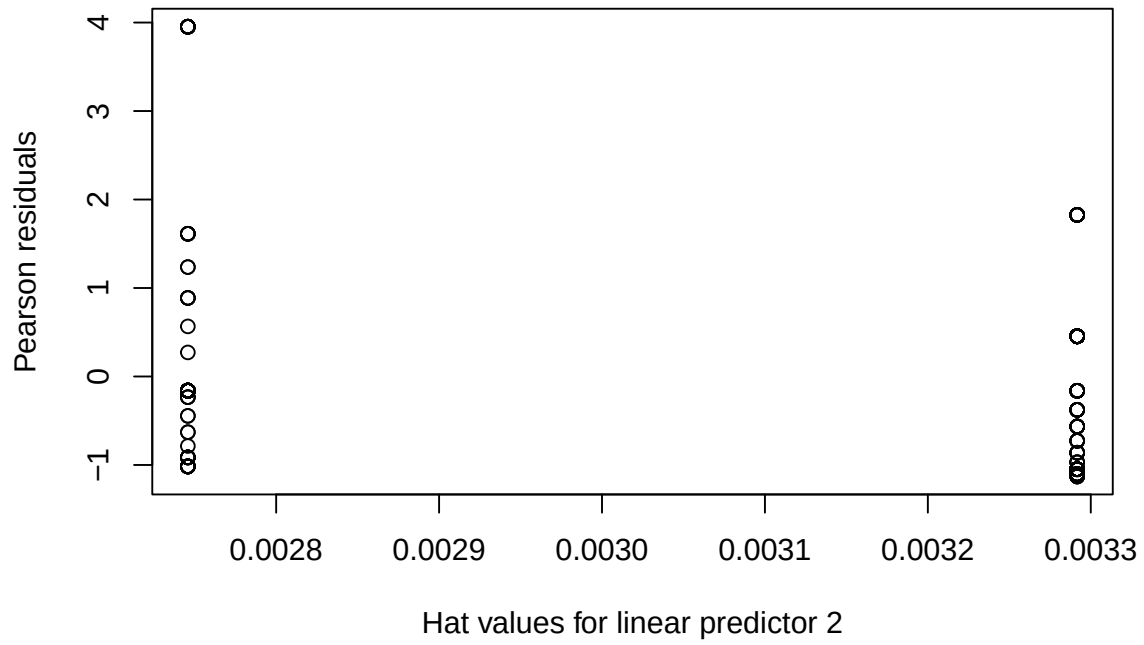
```
Call:
vglm(formula = accuracy ~ condition, family = tobit(Upper = 13),
    data = df)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept):1  -2.3913     0.8854   -2.70   0.0069 **
(Intercept):2   2.2155     0.0653   33.93   <2e-16 ***
condition121    5.9518     1.1742    5.07    4e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Names of linear predictors: mu, loglink(sd)

Log-likelihood: -677 on 657 degrees of freedom

Number of Fisher scoring iterations: 7

No Hauck-Donner effect found in any of the estimates
```

```r
plot(t)
```


Pearson residuals
−1.0
0.0
0.5
1.0
1.5
−2
−1
0
1
2
3
Linear predictor 1


Pearson residuals
4
3
2
1
0
−1
1.5
2.0
2.5
3.0
Jittered linear predictor 2


Pearson residuals
−1.0
0.0
0.5
1.0
1.5
0.0058
0.0059
0.0060
0.0061
0.0062
0.0063
Hat values for linear predictor 1



Model Comparison

```
compare_performance(lm.1, p.1, nb.1, zinb.1)
```

```
# Comparison of Model Performance Indices

Name   |    Model |      AIC | AIC weights |      BIC | BIC weights |  RMSE | Sigma | Score_
-------------------------------------------------------------------------------------------
lm.1   |       lm | 1699.782 |     < 0.001 | 1711.179 |     < 0.001 | 3.150 | 3.160 |
p.1    |      glm | 1955.788 |     < 0.001 | 1963.386 |     < 0.001 | 3.150 | 2.136 |    -2.
nb.1   |   negbin | 1194.129 |     < 0.001 | 1205.526 |     < 0.001 | 3.150 | 0.909 |    -2.
zinb.1 | zeroinfl | 1073.880 |        1.00 | 1092.876 |        1.00 | 3.150 | 3.174 |    -1.
```

For modelling test phase absolute score (# items correct) it seems that the zero inflated negative binomial model is the best fit according to R2 and AIC, however, I am not clear on the implications of the interpretation (non significant in count process, significant on logit process), and also not clear if # items correct is truly a count process.

```
#uncertainty model visualization
# df %>%
  # data_grid(pretty_condition) %>%
  # augment(m, newdata = ., se_fit = TRUE) %>%
  # ggplot(aes(y = pretty_condition)) +
  # stat_halfeye(
  #   aes(xdist = dist_student_t(df = df.residual(m),
  #       mu = .fitted, sigma = .se.fit)), scale = .5) +
  # # add raw data in too (scale = .5 above adjusts the halfeye height so
  # # that the data fit in as well)
  # geom_jitter(aes(x = x), data = df, pch = "|", size = 2,
  #             position =   position_nudge(y = -.15), alpha = 0.5) +
  # labs (title = "Model Estimates with Uncertainty", x = "model coefficient") +
  # theme_minimal()
```

HURLDE BETA Regression

https://github.com/markhwhiteii/beta_hurdle/blob/master/manuscript/beta_hurdle.pdf

```
library(gamlss)
```

```
Loading required package: gamlss.data
```

```
Attaching package: 'gamlss.data'
```

```
The following object is masked from 'package:datasets':

    sleep
```

```
Loading required package: gamlss.dist
```

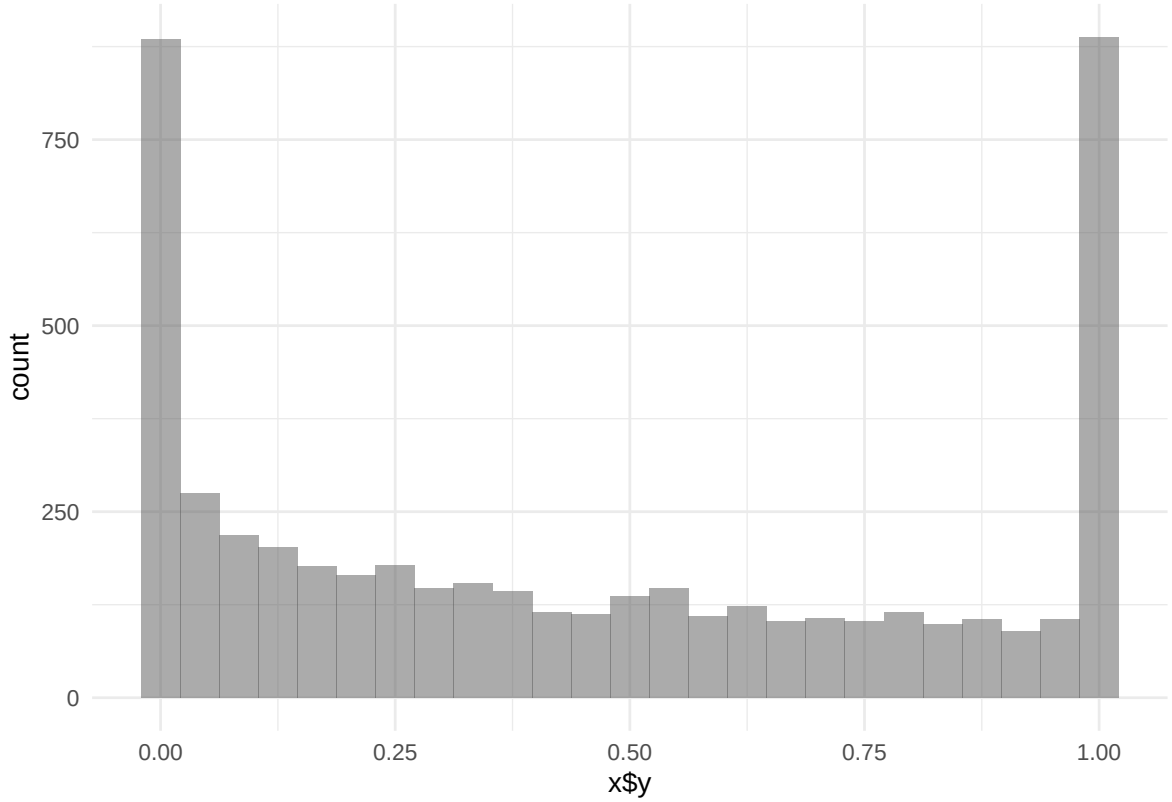
```
Loading required package: nlme
```

```
Attaching package: 'nlme'
```

```
The following object is masked from 'package:dplyr':

    collapse
```

```
The following object is masked from 'package:lme4':

    lmList

Loading required package: parallel

 **********   GAMLSS Version 5.4-3  **********

For more on GAMLSS look at https://www.gamlss.com/

Type gamlssNews() to see new features/changes/bug fixes.

Attaching package: 'gamlss'

The following object is masked from 'package:lme4':

    refit
```

```r
#CREATE SAMPLE DATA
n <- 5000
mu <- 0.40
sigma <- 0.60
p0 <- 0.13
p1 <- 0.17
p2 <- 1- p0- p1
a <- mu * (1- sigma ^ 2) / (sigma ^ 2)
b <- a * (1- mu) / mu

#CREATE DIST
set.seed(1839)
y <- rbeta(n, a, b)
cat <- sample(1:3, n, prob = c(p0, p2, p1), replace = TRUE)
y[cat == 1] <- 0
y[cat == 3] <- 1

#VISUALIZE DISTRIBUTION
x <- as.data.frame(y)
gf_histogram(~x$y)
```




```
#this looks not unlike my distribution!

#CREATE AN EMPTY MODEL
fit <- gamlss( formula = y ~ 1, # formula for mu
               formula.sigma = ~ 1, # formula for sigma
               formula.nu = ~ 1, # formula for nu
               formula.tau = ~ 1, # formula for tau
               family = BEINF() )
```

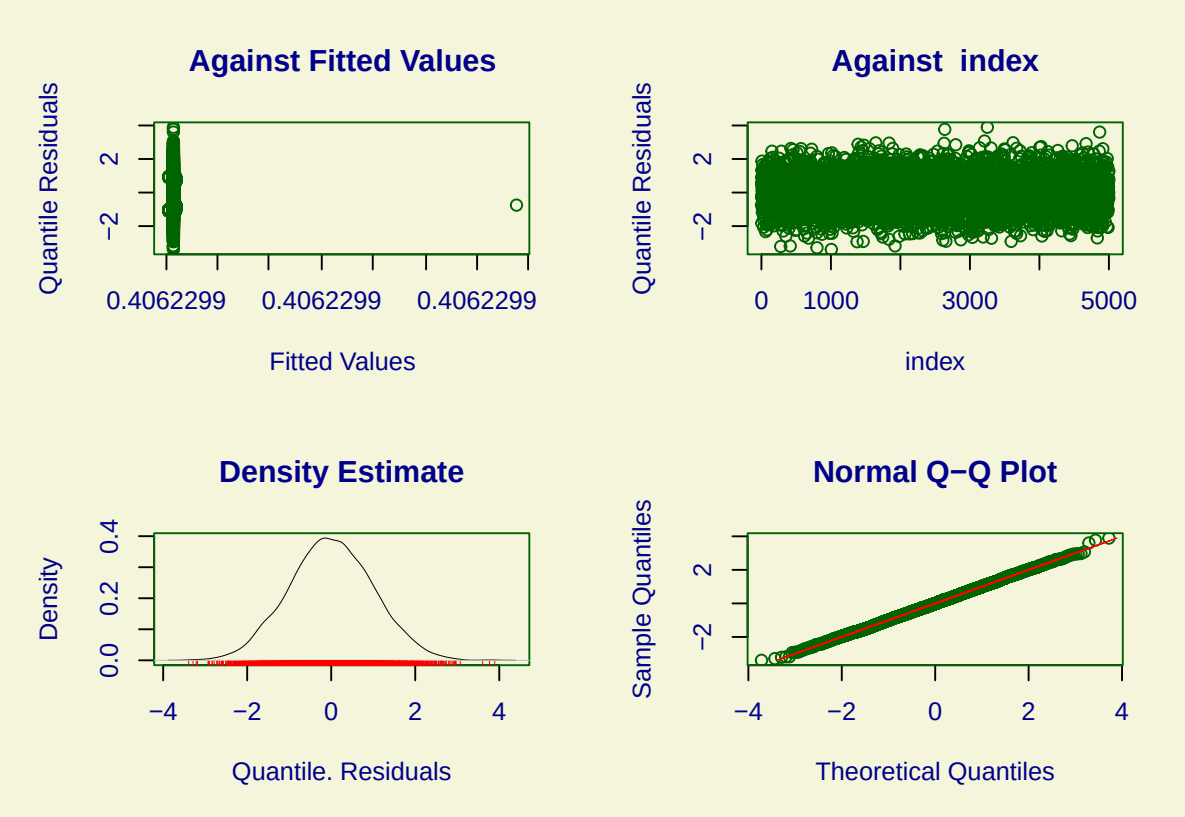
```
GAMLSS-RS iteration 1: Global Deviance = 7799
GAMLSS-RS iteration 2: Global Deviance = 7778
GAMLSS-RS iteration 3: Global Deviance = 7778
GAMLSS-RS iteration 4: Global Deviance = 7778
```

```
summary(fit)
```

```
******************************************************************
Family:  c("BEINF", "Beta Inflated")

Call:  gamlss(formula = y ~ 1, family = BEINF(), formula.sigma = ~1,
    formula.nu = ~1, formula.tau = ~1)

Fitting method: RS()

------------------------------------------------------------------
Mu link function:  logit
Mu Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.3796     0.0196   -19.4   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Sigma link function:  logit
Sigma Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.3951     0.0162    24.5   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Nu link function:  log
Nu Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.632      0.042   -38.9   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Tau link function:  log
Tau Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.4014     0.0382   -36.7   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
No. of observations in the fit:  5000
Degrees of Freedom for the fit:  4
```

```
      Residual Deg. of Freedom:  4996
                      at cycle:  4

Global Deviance:     7778
            AIC:     7786
            SBC:     7812
*******************************************************************
```

```
plot(fit)
```




```
*******************************************************************
     Summary of the Randomised Quantile Residuals
                           mean   =  0.000571
                       variance   =  1
                  coef. of skewness  =  0.0294
                  coef. of kurtosis  =  2.95
Filliben correlation coefficient  =  1
*******************************************************************
```

```r
#TRANSFORM PARAMETRS BACK
inv_logit <- function(x) exp(x) / (1 + exp(x)) # inverse of link function
fit_mu <- inv_logit(fit$mu.coefficients)
paste("MU: ",fit_mu)
```

```
[1] "MU:  0.406229902102452"
```

```r
fit_sigma <- inv_logit(fit$sigma.coefficients)
paste("SIGMA: ",fit_sigma)
```

```
[1] "SIGMA:  0.597499259410111"
```

```r
fit_nu <- exp(fit$nu.coefficients)
fit_tau <- exp(fit$tau.coefficients)
fit_p0 <- fit_nu / (1 + fit_nu + fit_tau)
paste("P0: ",fit_p0)
```

```
[1] "P0:  0.135600165493784"
```

```r
fit_p1 <- fit_tau / (1 + fit_nu + fit_tau)
paste("P1: ",fit_p1)
```

```
[1] "P1:  0.170800000002391"
```

BETA HURDLE INTERPRETATION - beta component
- MU "location" (mean)
- SIGMA "scale" (positively related to variance; variance = sigma.squared *mean* (1-mean)
- Rigby, Stasinopoulos, Heller, and De Bastiani (2017) "reparameterized" the beta distribution so that the two parameters determining the shape of the distribution would be more useful in a regression framework (see Ferrari & Cribari-Neto, 2004 for a different parameterization)

ZERO-ONE HURDLE COMPONENT
- The two additional parameters, NU and TAU , are related to p0 and p1, respectively.
- p0 is the probability that a case equals 0,
- p1 is the probability that a case equals 1,
- p2 (i.e., 1 −p0 −p1) is the probability that the case comes from the beta distribution

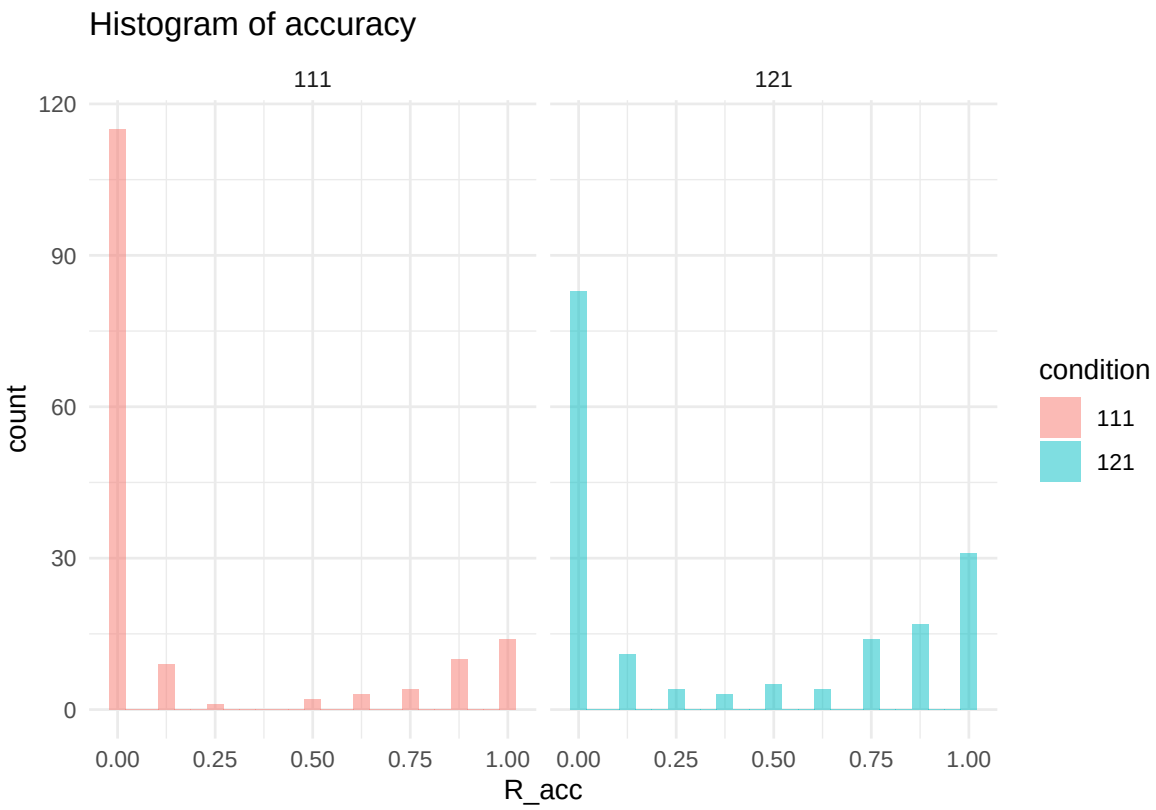
```r
#SETUP DATA

min = 0 #min possible value of scale
max = 8 #max possible value of scale

library(mosaic) #for shuffling
```

```
Registered S3 method overwritten by 'mosaic':
  method                           from
  fortify.SpatialPolygonsDataFrame ggplot2

The 'mosaic' package masks several functions from core packages in order to add
additional features.  The original behavior of these functions should not be affected by this

Attaching package: 'mosaic'

The following objects are masked from 'package:VGAM':

    chisq, logit

The following objects are masked from 'package:dplyr':

    count, do, tally

The following object is masked from 'package:purrr':

    cross

The following object is masked from 'package:lmerTest':

    rand

The following object is masked from 'package:lme4':

    factorize

The following object is masked from 'package:Matrix':

    mean
```

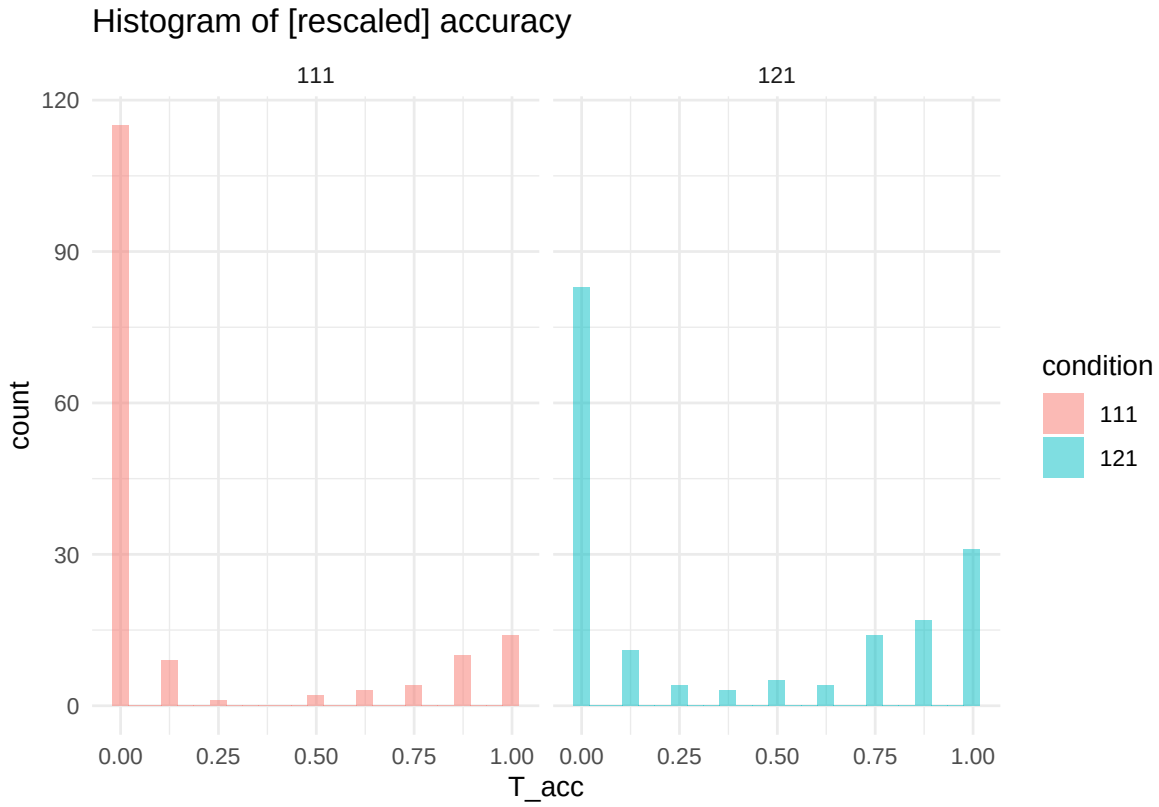
```
The following object is masked from 'package:modelr':

    resample

The following object is masked from 'package:vcd':

    mplot

The following object is masked from 'package:scales':

    rescale

The following object is masked from 'package:cowplot':

    theme_map

The following object is masked from 'package:ggplot2':

    stat

The following objects are masked from 'package:stats':

    binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
    quantile, sd, t.test, var

The following objects are masked from 'package:base':

    max, mean, min, prod, range, sample, sum
```

```r
#1. Rescale accuracy using
# recommended adjustment
#rescaled = value-min/(max-min)
df <- df_subjects %>% mutate(
  accuracy = item_test_NABS,
  R_acc = (accuracy-min)/(max-min), #as %
  T_acc = (accuracy * (nrow(df)-1) + 0.5)/nrow(df)/8, #transform for no 0 and 1
  perm = shuffle(condition),
  scaffold_rt = item_scaffold_rt
) %>% dplyr::select(accuracy,R_acc, T_acc, condition, perm,scaffold_rt)
```

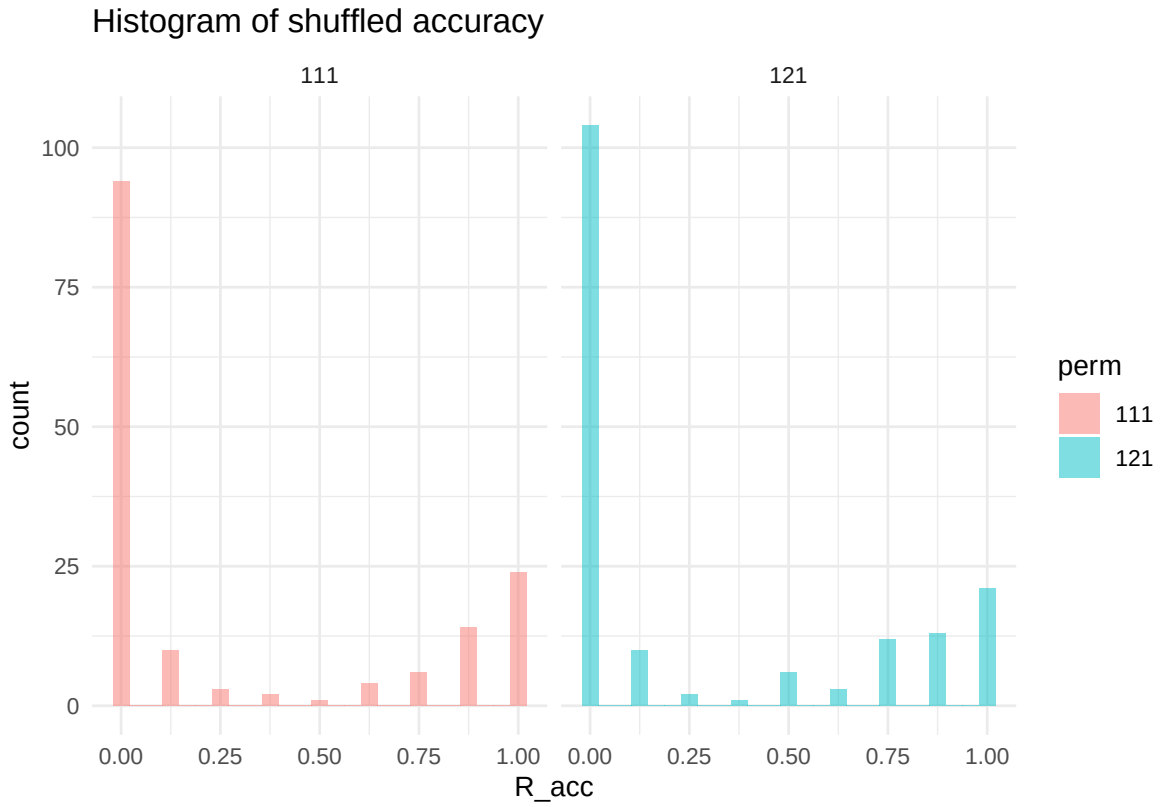
```
#VISUALIZE DISTRIBUTION
gf_histogram(~R_acc, fill = ~condition, data = df) %>% gf_facet_wrap(~condition) + labs(ti
```




```
#VISUALIZE DISTRIBUTION
gf_histogram(~T_acc, fill = ~condition, data = df) %>% gf_facet_wrap(~condition) + labs(ti
```




```
gf_histogram(~R_acc, fill = ~perm, data = df) %>% gf_facet_wrap(~perm) + labs(title = "His
```




```
#SUMMARIZE SAMPLE
paste("Grand mean", mean(df$R_acc))
```

```
[1] "Grand mean 0.2875"
```

```
library(mosaic)
stats = favstats(df$R_acc ~ df$condition)
stats$mean <- mean(df$R_acc ~ df$condition)
stats$var <- var(df$R_acc ~ df$condition)
print("Grand stats")
```

```
[1] "Grand stats"
```

```
stats
```

```
  df$condition min Q1 median    Q3 max  mean    sd   n missing   var
1          111   0  0  0.000 0.125   1 0.190 0.358 158       0 0.128
2          121   0  0  0.125 0.875   1 0.377 0.426 172       0 0.182
```

```
print("P0")
```

```
[1] "P0"
```

```
nrow(df %>% filter(R_acc ==0))/nrow(df)
```

```
[1] 0.6
```

```
print("P1")
```

```
[1] "P1"
```

```
nrow(df %>% filter(R_acc ==1))/nrow(df)
```

```
[1] 0.136
```

```
#CREATE MODEL

#CREATE AN EMPTY MODEL
m0 <- gamlss( formula = R_acc ~ 1, # formula for mu
              formula.sigma =  ~ 1, # formula for sigma
              formula.nu =  ~ 1, # formula for nu
              formula.tau =  ~ 1, # formula for tau
              family = BEINF(), data = df )
```

```
GAMLSS-RS iteration 1: Global Deviance = 610
GAMLSS-RS iteration 2: Global Deviance = 609
GAMLSS-RS iteration 3: Global Deviance = 609
GAMLSS-RS iteration 4: Global Deviance = 609
GAMLSS-RS iteration 5: Global Deviance = 609
```

```r
m0 <- gamlss(R_acc ~ 1, ~ 1, ~ 1, ~ 1,
             data = df, family = BEINF())
```

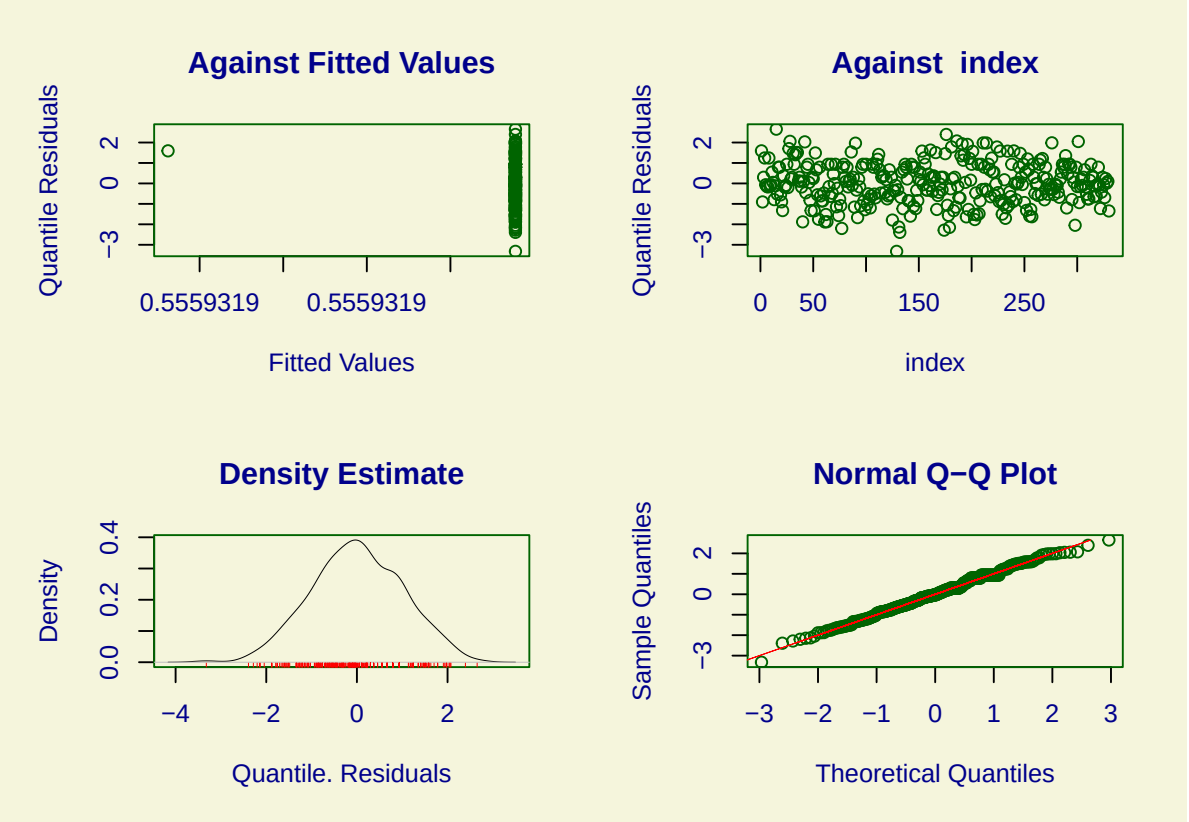
```
GAMLSS-RS iteration 1: Global Deviance = 610
GAMLSS-RS iteration 2: Global Deviance = 609
GAMLSS-RS iteration 3: Global Deviance = 609
GAMLSS-RS iteration 4: Global Deviance = 609
GAMLSS-RS iteration 5: Global Deviance = 609
```

```r
summary(m0)
```

```
******************************************************************
Family:  c("BEINF", "Beta Inflated")

Call:  gamlss(formula = R_acc ~ 1, sigma.formula = ~1, nu.formula = ~1,
    tau.formula = ~1, family = BEINF(), data = df)

Fitting method: RS()

------------------------------------------------------------------
Mu link function:  logit
Mu Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.225      0.113    1.98    0.048 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Sigma link function:  logit
Sigma Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.177      0.100    1.76    0.079 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Nu link function:  log
Nu Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.822      0.129    6.39  5.7e-10 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Tau link function:  log
Tau Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.660      0.184   -3.59  0.00038 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
No. of observations in the fit:  330
Degrees of Freedom for the fit:  4
      Residual Deg. of Freedom:  326
                      at cycle:  5

Global Deviance:     609
            AIC:     617
            SBC:     632
******************************************************************
```

```
plot(m0)
```




```
**********************************************************************
     Summary of the Randomised Quantile Residuals
                           mean   =  0.0109
                       variance   =  0.996
                  coef. of skewness  =  -0.0563
                  coef. of kurtosis  =  2.79
Filliben correlation coefficient  =  0.998
**********************************************************************
```

```r
#TRANSFORM PARAMETRS BACK
inv_logit <- function(x) exp(x) / (1 + exp(x)) # inverse of link function
m0_mu <- inv_logit(m0$mu.coefficients)
paste("MU: ",m0_mu)
```

```
[1] "MU:  0.555931874775555"
```

```r
m0_sigma <- inv_logit(m0$sigma.coefficients)
paste("SIGMA: ",m0_sigma)
```

```
[1] "SIGMA:  0.544134514840075"
```

```r
m0_nu <- exp(m0$nu.coefficients)
paste("NU: ",m0_nu)
```

```
[1] "NU:  2.27484150905293"
```

```r
m0_tau <- exp(m0$tau.coefficients)
paste("TAU: ",m0_tau)
```

```
[1] "TAU:  0.517080427912532"
```

```r
m0_p0 <- fit_nu / (1 + fit_nu + fit_tau)
paste("P0: ",m0_p0)
```

```
[1] "P0:  0.135600165493784"
```

```r
m0_p1 <- fit_tau / (1 + fit_nu + fit_tau)
paste("P1: ",m0_p1)
```

```
[1] "P1:  0.170800000002391"
```

```r
#CREATE PREDICTOR MODEL
m1 <- gamlss(R_acc ~ condition, ~ condition, ~ condition, ~ condition,
             data = df, family = BEINF())
```

```
GAMLSS-RS iteration 1: Global Deviance = 588
GAMLSS-RS iteration 2: Global Deviance = 587
GAMLSS-RS iteration 3: Global Deviance = 587
GAMLSS-RS iteration 4: Global Deviance = 587
GAMLSS-RS iteration 5: Global Deviance = 587
```

```r
summary(m1)
```

```
******************************************************************
Family:  c("BEINF", "Beta Inflated")

Call:  gamlss(formula = R_acc ~ condition, sigma.formula = ~condition,
    nu.formula = ~condition, tau.formula = ~condition,
    family = BEINF(), data = df)

Fitting method: RS()

------------------------------------------------------------------
Mu link function:  logit
Mu Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.144      0.203    0.71     0.48
condition121    0.124      0.244    0.51     0.61

------------------------------------------------------------------
Sigma link function:  logit
Sigma Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.280      0.174    1.61     0.11
condition121   -0.164      0.213   -0.77     0.44

------------------------------------------------------------------
Nu link function:  log
Nu Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.377      0.208    6.63  1.4e-10 ***
condition121   -1.019      0.269   -3.79  0.00018 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Tau link function:  log
Tau Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.729      0.325   -2.24    0.026 *
condition121    0.102      0.394    0.26    0.796
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
------------------------------------------------------------------
No. of observations in the fit:  330
Degrees of Freedom for the fit:  8
      Residual Deg. of Freedom:  322
                      at cycle:  5

Global Deviance:     587
            AIC:     603
            SBC:     633
******************************************************************
```

```r
#LOOKING PREDICTOR MODEL
m <- gamlss(R_acc ~ condition ,
            ~ condition ,
            ~ condition ,
            data = df, family = BEINF())
```

```
GAMLSS-RS iteration 1: Global Deviance = 588
GAMLSS-RS iteration 2: Global Deviance = 587
GAMLSS-RS iteration 3: Global Deviance = 587
GAMLSS-RS iteration 4: Global Deviance = 587
GAMLSS-RS iteration 5: Global Deviance = 587
```

```r
summary(m)
```

```
******************************************************************
Family:  c("BEINF", "Beta Inflated")

Call:  gamlss(formula = R_acc ~ condition, sigma.formula = ~condition,
    nu.formula = ~condition, tau.formula = ~condition,
    family = BEINF(), data = df)

Fitting method: RS()

------------------------------------------------------------------
Mu link function:  logit
Mu Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)     0.144      0.203    0.71     0.48
condition121    0.124      0.244    0.51     0.61

-------------------------------------------------------------------
Sigma link function:  logit
Sigma Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.280      0.174    1.61     0.11
condition121   -0.164      0.213   -0.77     0.44

-------------------------------------------------------------------
Nu link function:  log
Nu Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.377      0.208    6.63  1.4e-10 ***
condition121   -1.019      0.269   -3.79  0.00018 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

-------------------------------------------------------------------
Tau link function:  log
Tau Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.729      0.325   -2.24    0.026 *
condition121    0.102      0.394    0.26    0.796
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

-------------------------------------------------------------------
No. of observations in the fit:  330
Degrees of Freedom for the fit:  8
      Residual Deg. of Freedom:  322
                      at cycle:  5

Global Deviance:     587
            AIC:     603
            SBC:     633
*******************************************************************
```

```r
#CREATE PREDICTOR MODEL ON SHUFFLED [PERMUTATION TEST]
mperm <- gamlss(R_acc ~ perm, ~ perm, ~ perm, ~ perm,
            data = df, family = BEINF())
```

```
GAMLSS-RS iteration 1: Global Deviance = 609
GAMLSS-RS iteration 2: Global Deviance = 608
GAMLSS-RS iteration 3: Global Deviance = 608
GAMLSS-RS iteration 4: Global Deviance = 608
GAMLSS-RS iteration 5: Global Deviance = 608
```

```
summary(mperm)
```

```
******************************************************************
Family:  c("BEINF", "Beta Inflated")

Call:  gamlss(formula = R_acc ~ perm, sigma.formula = ~perm,
    nu.formula = ~perm, tau.formula = ~perm, family = BEINF(),      data = df)


Fitting method: RS()

------------------------------------------------------------------
Mu link function:  logit
Mu Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.1914     0.1706    1.12     0.26
perm121       0.0635     0.2278    0.28     0.78

------------------------------------------------------------------
Sigma link function:  logit
Sigma Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.237      0.148    1.60     0.11
perm121       -0.117      0.202   -0.58     0.56

------------------------------------------------------------------
Nu link function:  log
Nu Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8535     0.1887    4.52  8.6e-06 ***
perm121      -0.0595     0.2579   -0.23     0.82
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
```

```
Tau link function:  log
Tau Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.511      0.258   -1.98    0.048 *
perm121       -0.294      0.368   -0.80    0.425
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
No. of observations in the fit:  330
Degrees of Freedom for the fit:  8
      Residual Deg. of Freedom:  322
                      at cycle:  5

Global Deviance:     608
            AIC:     624
            SBC:     654
******************************************************************
```

```
#sanity check with scaled outcome, no zeros ones
m3 <- gamlss(T_acc ~ condition, ~ condition, ~ condition, ~ condition,
             data = df, family = BEINF())
```

```
GAMLSS-RS iteration 1: Global Deviance = -1812
GAMLSS-RS iteration 2: Global Deviance = -2024
GAMLSS-RS iteration 3: Global Deviance = -2038
GAMLSS-RS iteration 4: Global Deviance = -2040
GAMLSS-RS iteration 5: Global Deviance = -2040
GAMLSS-RS iteration 6: Global Deviance = -2040
GAMLSS-RS iteration 7: Global Deviance = -2040
GAMLSS-RS iteration 8: Global Deviance = -2040
GAMLSS-RS iteration 9: Global Deviance = -2040
```

```
summary(m3)
```

```
Warning in summary.gamlss(m3): summary: vcov has failed, option qr is used instead

******************************************************************
Family:  c("BEINF", "Beta Inflated")
```

```
Call:  gamlss(formula = T_acc ~ condition, sigma.formula = ~condition,
    nu.formula = ~condition, tau.formula = ~condition, family = BEINF(),
    data = df)

Fitting method: RS()

------------------------------------------------------------------
Mu link function:  logit
Mu Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.1492     0.0981  -11.71  < 2e-16 ***
condition121   0.5677     0.1411    4.02 0.000071 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Sigma link function:  logit
Sigma Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.4399     0.0745   19.33   <2e-16 ***
condition121   0.1694     0.1038    1.63      0.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Nu link function:  log
Nu Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.25e+01   3.78e+03   -0.01        1
condition121 -6.72e-15   5.24e+03    0.00        1

------------------------------------------------------------------
Tau link function:  log
Tau Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.26e+01   3.96e+03   -0.01        1
condition121  9.20e-15   5.48e+03    0.00        1

------------------------------------------------------------------
No. of observations in the fit:  330
Degrees of Freedom for the fit:  8
      Residual Deg. of Freedom:  322
                      at cycle:  9
```

```
Global Deviance:     -2040
            AIC:     -2024
            SBC:     -1994
****************************************************************
```

```
#m3 shouldn't show condition as significant for nu and tau, because T_acc was scaled to no

#investigate beta negative binomial distribution
#https://en.wikipedia.org/wiki/Beta_negative_binomial_distribution

#TRANSFORM PARAMETRS BACK
inv_logit <- function(x) exp(x) / (1 + exp(x)) # inverse of link function
m1_mu <- inv_logit(m1$mu.coefficients)
paste("MU: ",m1_mu)
```

```
[1] "MU:  0.536038311159578" "MU:  0.531024352873784"
```

```
m1_sigma <- inv_logit(m0$sigma.coefficients)
paste("SIGMA: ",m1_sigma)
```

```
[1] "SIGMA:  0.544134514840075"
```

```
m1_nu <- exp(m1$nu.coefficients)
paste("NU: ",m1_nu)
```

```
[1] "NU:  3.96329406964311"  "NU:  0.360974858677686"
```

```
m1_tau <- exp(m1$tau.coefficients)
paste("TAU: ",m1_tau)
```

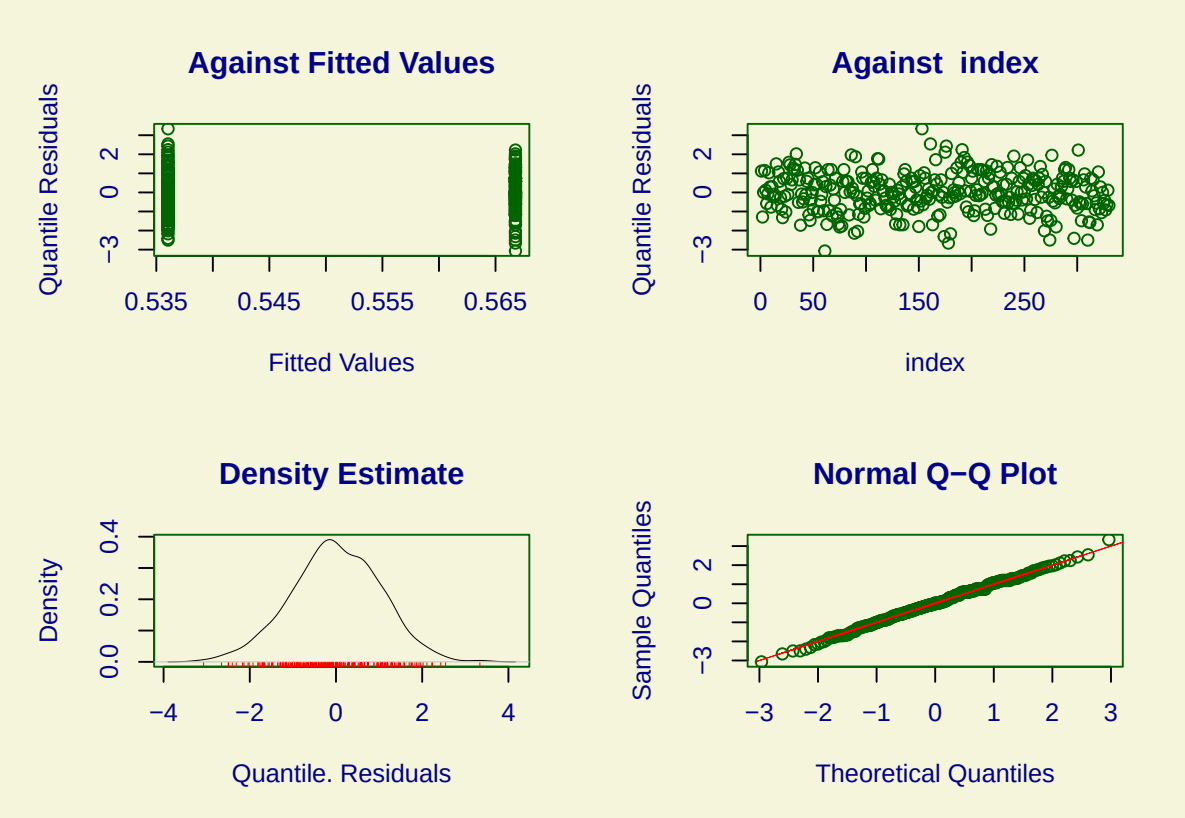
```
[1] "TAU:  0.482542801248665" "TAU:  1.10746276561553"
```

```
summary(m)
```

```
******************************************************************
Family:  c("BEINF", "Beta Inflated")

Call:  gamlss(formula = R_acc ~ condition, sigma.formula = ~condition,
    nu.formula = ~condition, tau.formula = ~condition,
    family = BEINF(), data = df)

Fitting method: RS()

------------------------------------------------------------------
Mu link function:  logit
Mu Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.144      0.203    0.71     0.48
condition121    0.124      0.244    0.51     0.61

------------------------------------------------------------------
Sigma link function:  logit
Sigma Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.280      0.174    1.61     0.11
condition121   -0.164      0.213   -0.77     0.44

------------------------------------------------------------------
Nu link function:  log
Nu Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.377      0.208    6.63  1.4e-10 ***
condition121   -1.019      0.269   -3.79  0.00018 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
Tau link function:  log
Tau Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.729      0.325   -2.24    0.026 *
condition121    0.102      0.394    0.26    0.796
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

------------------------------------------------------------------
No. of observations in the fit:  330
```

```
Degrees of Freedom for the fit:  8
      Residual Deg. of Freedom:  322
                       at cycle:  5

Global Deviance:     587
            AIC:     603
            SBC:     633
******************************************************************
```

```
plot(m)
```




```
******************************************************************
     Summary of the Randomised Quantile Residuals
                           mean   =  -0.00363
                       variance   =  1.04
                  coef. of skewness  =  -0.0633
                  coef. of kurtosis  =  3.03
```

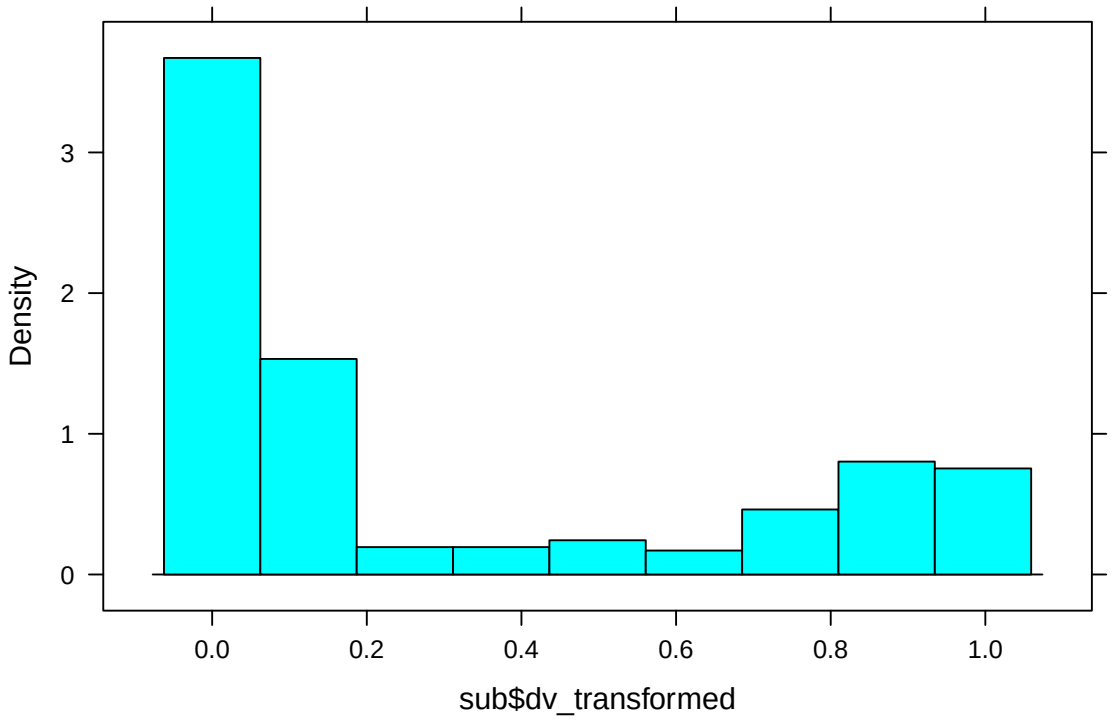
```
Filliben correlation coefficient  =  0.999
*****************************************************************
```

- MU tells if mean is different by condition
- SIGMA tells if variance is different by condition
- NU coefficient tells if condition yields different probability at floor
- TAU coefficient tells if condition yields different probability at ceiling

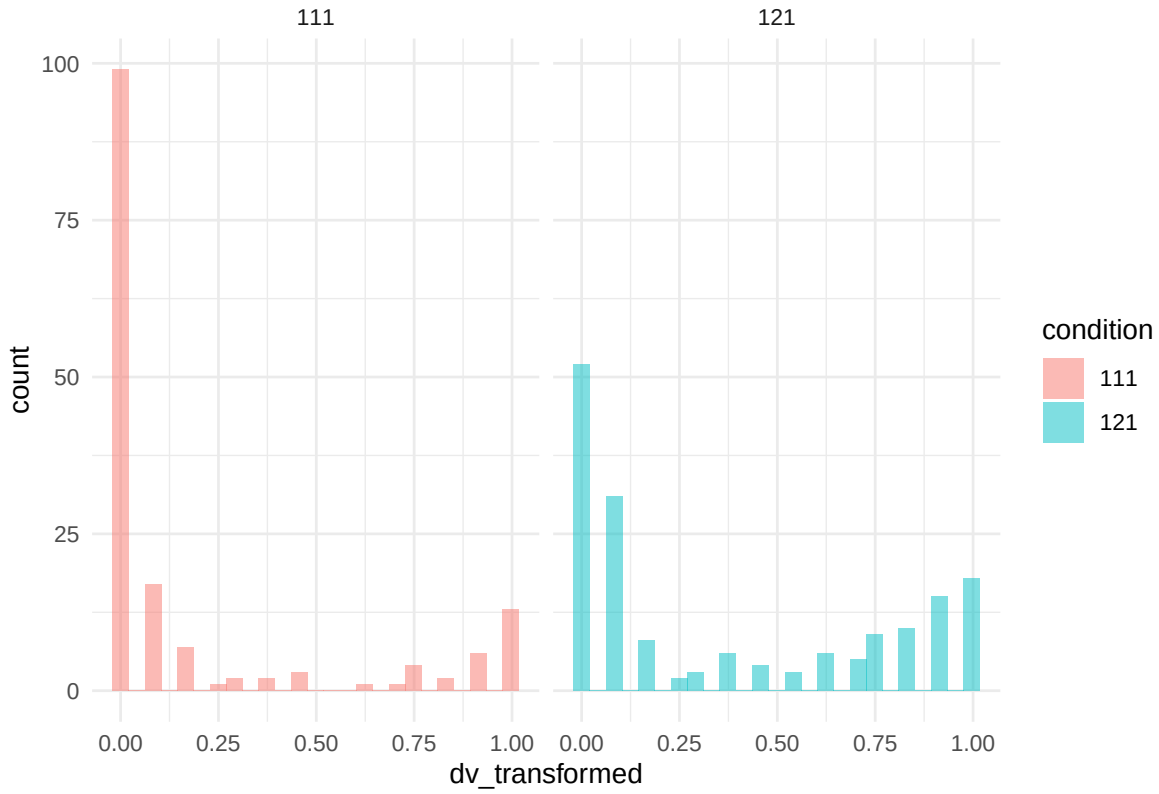
Beta Regression (% Correct)

Beta regression on % correct (with standard transformation for including [0,1]) https://stats.stackexchange.com/
to-interpret-the-coefficients-from-a-beta-regression

```
#
library(betareg)

#RESCLAE VARIABLE
#beta reg can't handle 0s and 1s
sub <- df_subjects %>% dplyr::select(condition, DV_percent_NABS)
n = nrow(sub) %>% unlist()
sub$dv_transformed = (sub$DV_percent_NABS * (n-1) + 0.5)/n

#VISUALIZE VARIABLES
histogram(sub$dv_transformed)
```




```
gf_histogram(~dv_transformed, fill = ~condition, data = sub) %>% gf_facet_wrap(~condition)
```

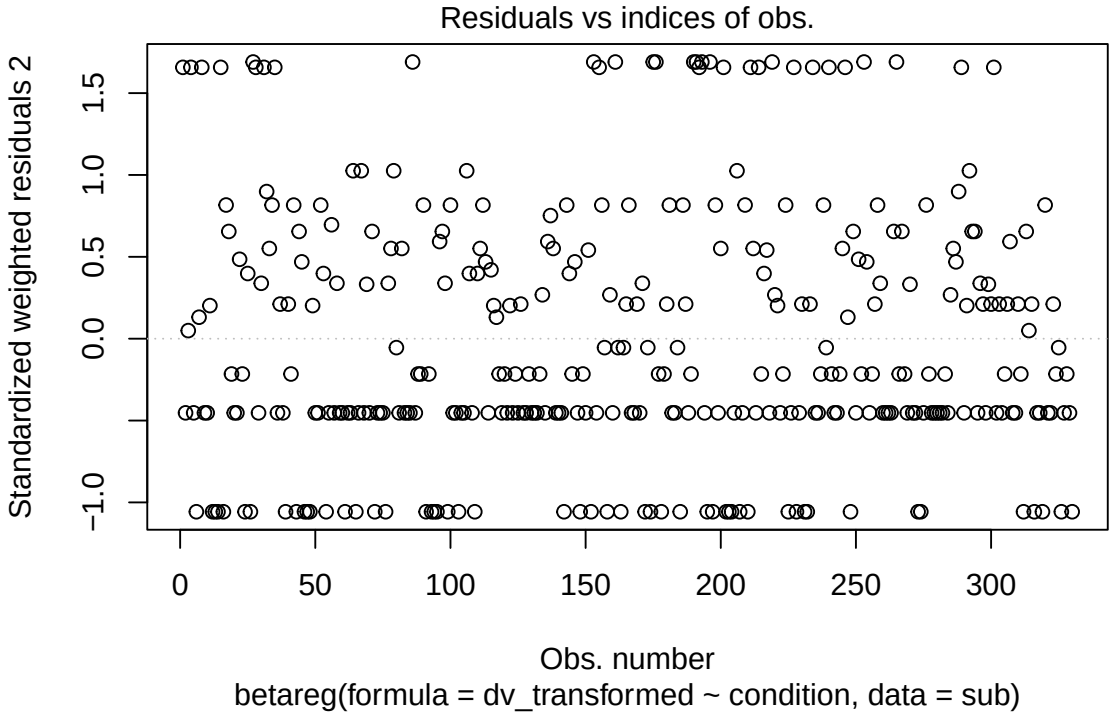


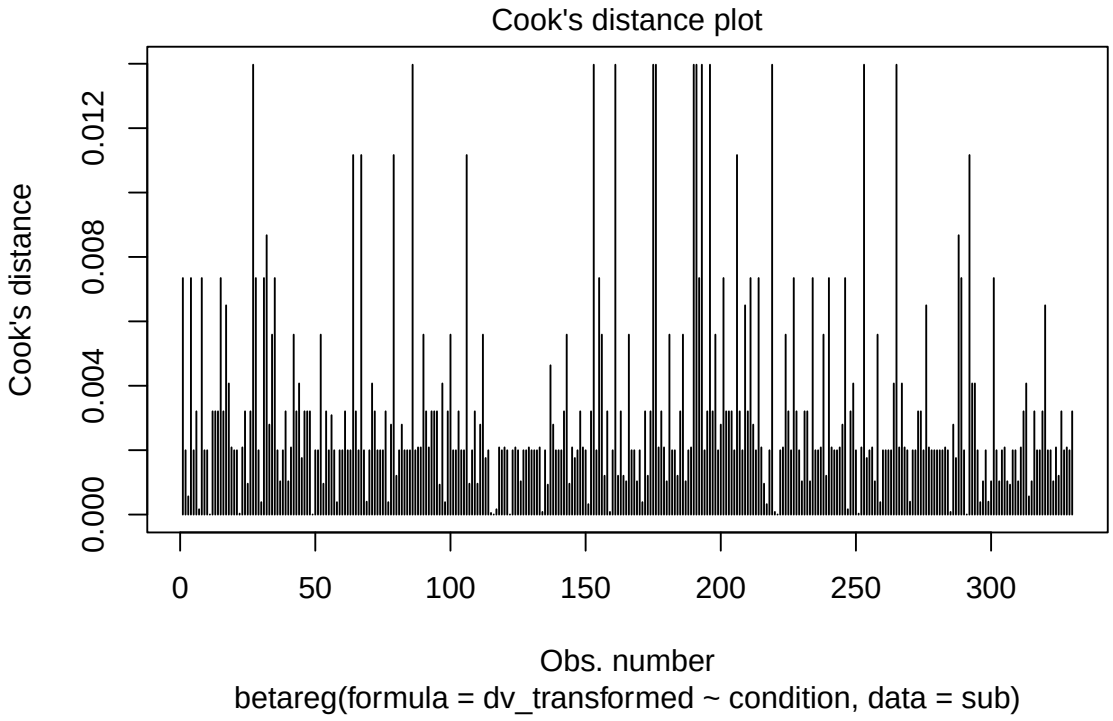

```
#FIT MODEL
mb <- betareg(dv_transformed ~ condition, data = sub)
summary(mb)
```

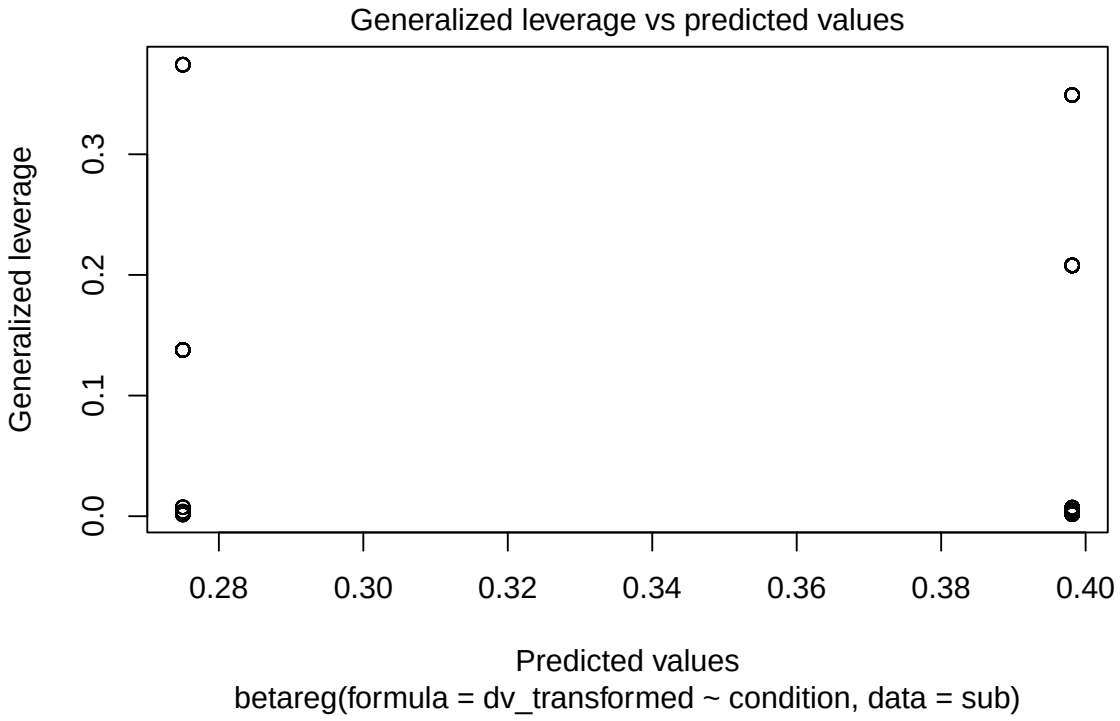
```
Call:
betareg(formula = dv_transformed ~ condition, data = sub)

Standardized weighted residuals 2:
   Min     1Q Median     3Q    Max
-1.057 -0.453 -0.216  0.541  1.690

Coefficients (mean model with logit link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.969      0.108   -8.97   <2e-16 ***
condition121    0.556      0.143    3.89   0.0001 ***

Phi coefficients (precision model with identity link):
      Estimate Std. Error z value Pr(>|z|)
```

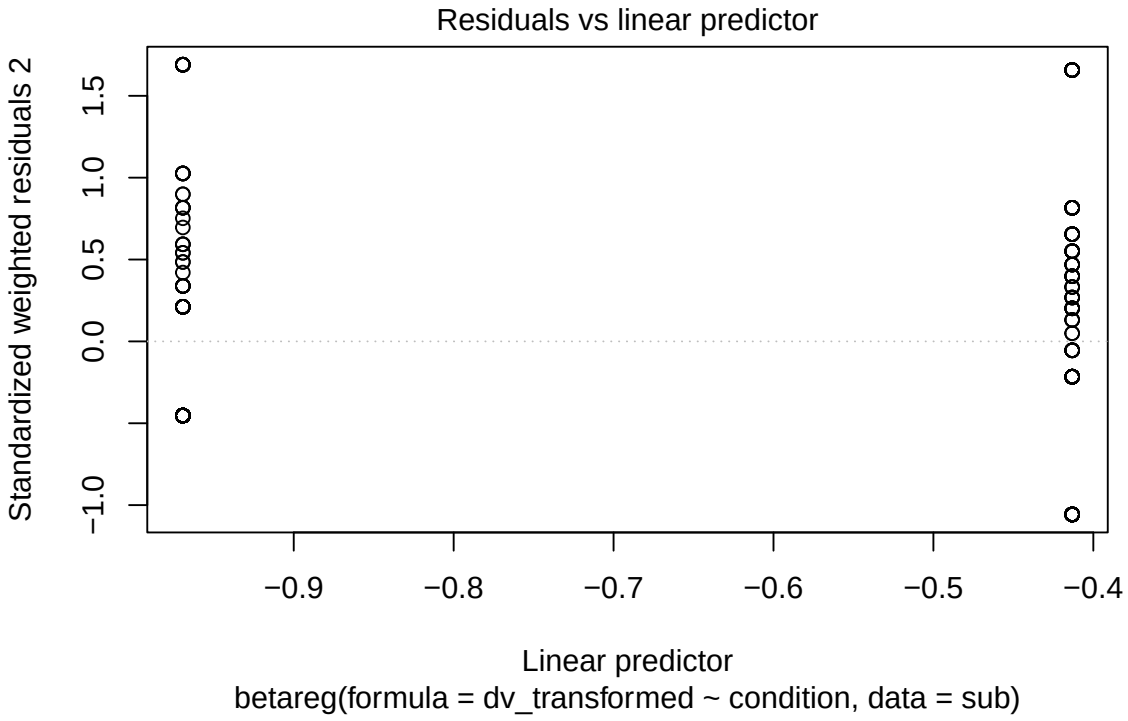
```
(phi)   0.6604     0.0425    15.5   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Type of estimator: ML (maximum likelihood)
Log-likelihood:  506 on 3 Df
Pseudo R-squared: 0.0725
Number of iterations: 12 (BFGS) + 1 (Fisher scoring)
```

```r
plot(mb)
```




Cook's distance plot
Cook's distance
0.000
0.004
0.008
0.012
0
50
100
150
200
250
300
Obs. number
betareg(formula = dv_transformed ~ condition, data = sub)


Generalized leverage vs predicted values
Generalized leverage
0.0
0.1
0.2
0.3
0.28
0.30
0.32
0.34
0.36
0.38
0.40
Predicted values
betareg(formula = dv_transformed ~ condition, data = sub)



WIP | HURDLE MODEL

- https://data.library.virginia.edu/getting-started-with-hurdle-models/
- https://en.wikipedia.org/wiki/Hurdle_model#:~:text=A%20hurdle%20model%20is%20a,of%20the%20nc

class of models for count data with both overdispersion and excess zeros;
different from zero-inflated models where the excess zeros are theorized to arise from two different processes; in the hurdle model, there is a model for P(x=0) and a separate model for P(x!=0)

The model includes: - A binary logit model to model whether the observation takes a positive count or not. - a truncated Poisson or Negative binomial model that only fits positive counts

This allows us to model: (1) Does the student get *any* questions right? (2) How many questions does the student get right?

```
library(pscl) #zero-inf and hurdle models
library(countreg) #rootogram
```

```
Registered S3 methods overwritten by 'countreg':
  method                 from
  print.zeroinfl         pscl
  print.summary.zeroinfl pscl
  summary.zeroinfl       pscl
  coef.zeroinfl          pscl
  vcov.zeroinfl          pscl
  logLik.zeroinfl        pscl
  predict.zeroinfl       pscl
  residuals.zeroinfl     pscl
  fitted.zeroinfl        pscl
  terms.zeroinfl         pscl
  model.matrix.zeroinfl  pscl
  extractAIC.zeroinfl    pscl
  print.hurdle           pscl
  print.summary.hurdle   pscl
  summary.hurdle         pscl
  coef.hurdle            pscl
  vcov.hurdle            pscl
  logLik.hurdle          pscl
  predict.hurdle         pscl
  residuals.hurdle       pscl
  fitted.hurdle          pscl
  terms.hurdle           pscl
  model.matrix.hurdle    pscl
  extractAIC.hurdle      pscl

Attaching package: 'countreg'

The following objects are masked from 'package:VGAM':

    dzipois, pzipois, qzipois, rzipois

The following objects are masked from 'package:pscl':

    hurdle, hurdle.control, hurdletest, zeroinfl, zeroinfl.control

The following object is masked from 'package:vcd':

    rootogram
```

```r
#install.packages("countreg", repos="http://R-Forge.R-project.org")

#SYNTAX OUTCOME ~ count model predictor | hurdle predictor

h.1 <- pscl::hurdle(item_test_NABS ~ condition | condition , data = df_subjects,
              zero.dist = "binomial", dist = "poisson", size = 8)
```

```
Warning in optim(fn = countDist, gr = countGrad, par = c(start$count, if (dist
== : unknown names in control: size
```

```
Warning in optim(fn = zeroDist, gr = zeroGrad, par = c(start$zero, if (zero.dist
== : unknown names in control: size
```

```r
h.2 <- pscl::hurdle(item_test_NABS ~ condition | condition , data = df_subjects,
              zero.dist = "binomial", dist = "negbin", size = 8)
```

```
Warning in optim(fn = countDist, gr = countGrad, par = c(start$count, if (dist
== : unknown names in control: size

Warning in optim(fn = countDist, gr = countGrad, par = c(start$count, if (dist
== : unknown names in control: size
```

```r
summary(h.1)
```

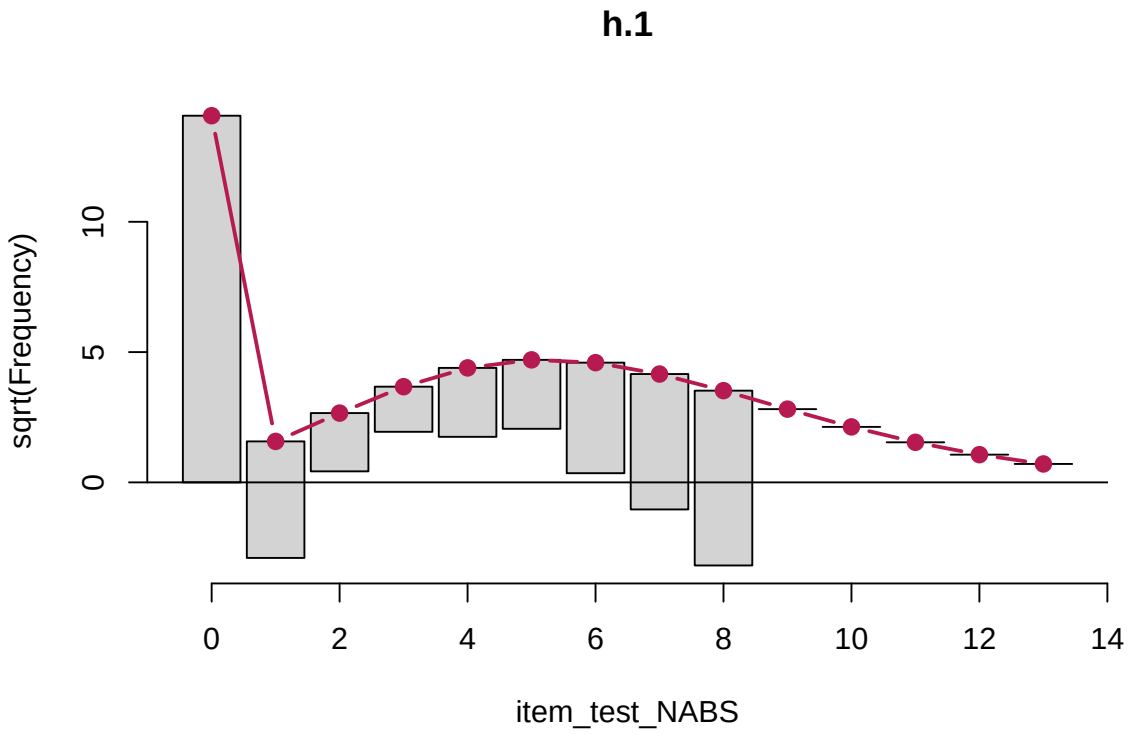
```
Call:
pscl::hurdle(formula = item_test_NABS ~ condition | condition, data = df_subjects,
    dist = "poisson", zero.dist = "binomial", size = 8)

Pearson residuals:
   Min     1Q Median     3Q    Max
-0.892 -0.818 -0.549  0.881  2.342

Count model coefficients (truncated poisson with log link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.7156     0.0653   26.29   <2e-16 ***
condition121   0.0447     0.0789    0.57     0.57
Zero hurdle model coefficients (binomial with logit link):
             Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)    -0.984      0.179   -5.50  3.7e-08 ***
condition121    1.054      0.235    4.48  7.4e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 10
Log-likelihood: -533 on 4 Df
```

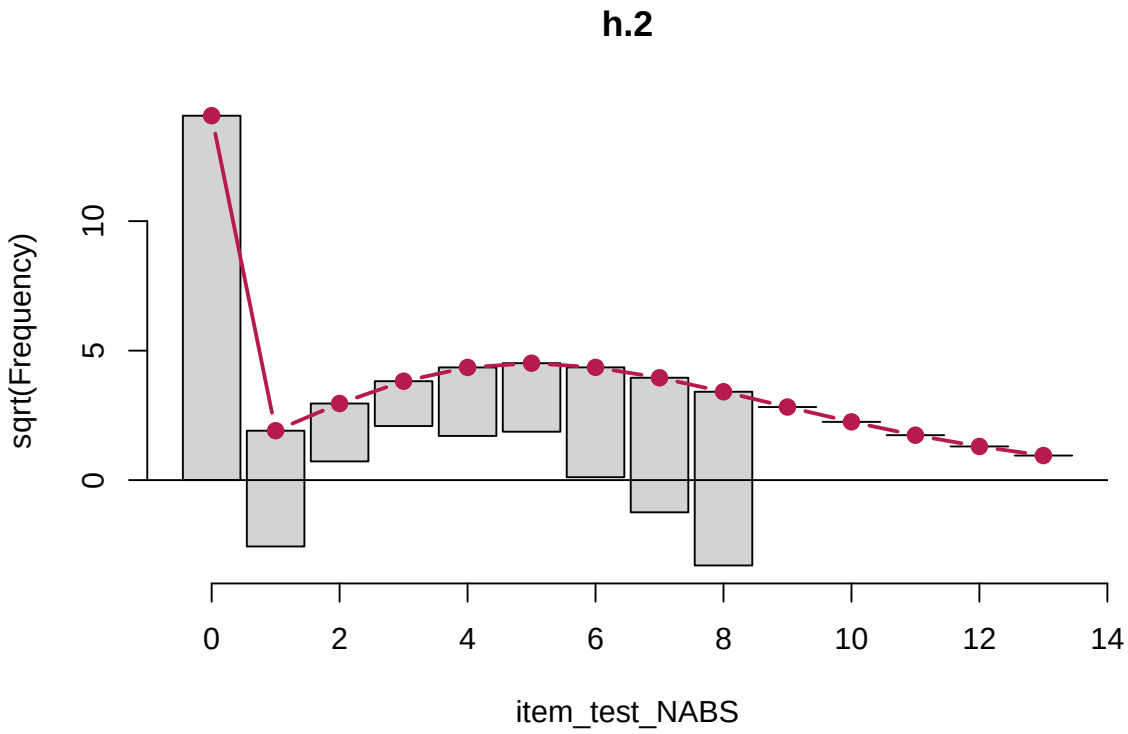
```r
summary(h.2)
```

```
Call:
pscl::hurdle(formula = item_test_NABS ~ condition | condition, data = df_subjects,
    dist = "negbin", zero.dist = "binomial", size = 8)

Pearson residuals:
   Min     1Q Median     3Q    Max
-0.866 -0.794 -0.538  0.856  2.294

Count model coefficients (truncated negbin with log link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.7126     0.0728   23.54  < 2e-16 ***
condition121   0.0451     0.0880    0.51  0.60810
Log(theta)     3.1851     0.8732    3.65  0.00026 ***
Zero hurdle model coefficients (binomial with logit link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.984      0.179   -5.50  3.7e-08 ***
condition121    1.054      0.235    4.48  7.4e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta: count = 24.169
Number of iterations in BFGS optimization: 20
Log-likelihood: -532 on 5 Df
```

```r
rootogram(h.1)
```




```
rootogram(h.2)
```




```
compare_performance(h.1,h.2)
```

```
Some of the nested models seem to be identical

# Comparison of Model Performance Indices

Name |  Model |      AIC | AIC weights |      BIC | BIC weights |    R2 | R2 (adj.) |  RMSE
-------------------------------------------------------------------------------------------
h.1  | hurdle | 1073.583 |       0.537 | 1088.780 |       0.886 | 0.351 |     0.347 | 3.150
h.2  | hurdle | 1073.880 |       0.463 | 1092.876 |       0.114 | 0.363 |     0.359 | 3.150
```

WIP UNKNOWN

Cummulative Ordinal (Bayesian)

https://journals.sagepub.com/doi/full/10.1177/2515245918823199

```
# library(brms)


# #DEFINE DATA
# df <- df_items %>% mutate(
#   scaled = factor(score_SCALED, ordered = TRUE,  #ordered factor
#                     levels = c("-1", "-0.5", "0", "0.5","1"))
# )
#
# ord_cum <- brm( formula = scaled ~ condition,
#                 data = df,
#                 family = cumulative("probit"),
#                 file = "analysis/SGC3A/models/m_items_ord.cum.rds" # cache model (can be
#
# )
#
# summary(ord_cum)
# conditional_effects(ord_cum, "condition", categorical = TRUE)
#
# #SJPLOT
# library(sjPlot)
# plot_model(ord_cum)
#
# # m %>%
# #   spread_draws(b_Intercept, r_condition[condition,]) %>%
# #   mutate(condition_mean = b_Intercept + r_condition) %>%
# #   ggplot(aes(y = condition, x = condition_mean)) +
# #   stat_halfeye()
#
# # performance(ord_cum)
# # plot(ord_cum)
```

Adjacent-Category Ordinal (Bayesian)

```
#
# #DEFINE DATA
# df <- df_items %>% mutate(
#   scaled = factor(score_SCALED, ordered = TRUE,  #ordered factor
#                     levels = c("-1", "-0.5", "0", "0.5","1"))
# )
```

```
#
#
# # To specify an adjacent-category model, we use family = acat() instead of family = cumulat
#
# ord_acat <- brm( formula = scaled ~ cs(condition),
#                 data = df,
#                 family = acat("probit"),
#                 file = "analysis/SGC3A/models/m_items_ord.acat.rds" # cache model (can be
# )
#
# summary(ord_acat)
# conditional_effects(ord_cum, "condition", categorical = TRUE)
# conditional_effects(ord_acat, "condition", categorical = TRUE)
#
# #TIDYBAYES VISUALIZATION
# library(tidybayes)
# ord_acat %>%
#   spread_draws(b_Intercept, r_condition[condition,]) %>%
#   mutate(condition_mean = b_Intercept + r_condition) %>%
#   ggplot(aes(y = condition, x = condition_mean)) +
#   stat_halfeye()
#
```

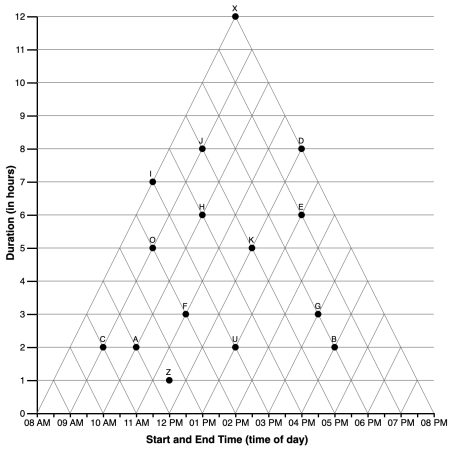
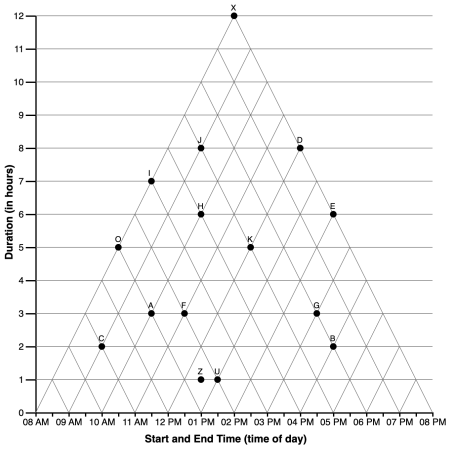
Part II

SGC3A

1 Introduction

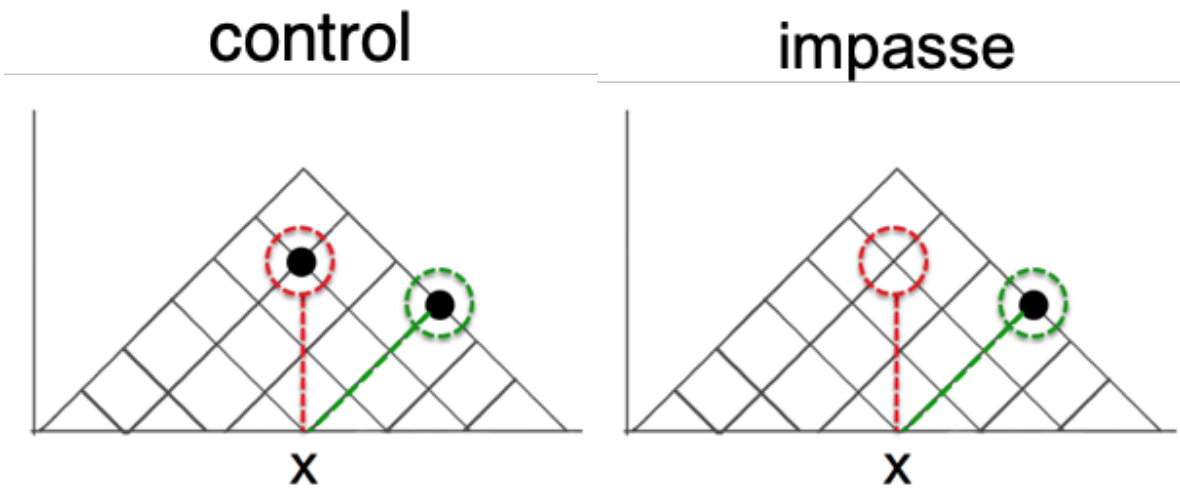
In Study 3A we explore the extent to which confronting a learner with an implicit obstacle (a mental impasse) influences their interpretation of the underlying coordinate system. This is a hypothesis that emerged from analysis of Study 2, leading us to suspect that *presenting a learner with a situation 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.*

Table 1.1: **SGC3A Study Conditions**

	**Control-Condition** Demo: 111
	**Impasse-Condition** Demo: 121

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.

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

#set some output options
```



(a) (b)

Figure 1.1: Posing a mental impasse

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

```
# HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# # mbp = "/Users/amyfox/Sites/RESEARCH/SGC-Scaffolding Graph Comprehension/SGC-X/ANALYSIS
# setwd(imac)

#IMPORT DATA
df_subjects <- read_rds('analysis/SGC3A/data/1-study-level/sgc3a_participants.rds')
```

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

1.0.1 Hypotheses

Experimental Hypothesis
Learners posed with scenario designed to evoke a mental impasse will be more likely to correct interpret the graph.

Table 1.2: Participants by Condition and Data Collection Period

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

- H1A | Learners in the IMPASSE condition will be more likely to correctly answer the first question than learners in CONTROL.
- H1B | Learners in the IMPASSE condition will score higher on the TEST Phase than learners in CONTROL.

Null Hypothesis
No significant differences in performance will exist between learners in the IMPASSE and CONTROL conditions.

Exploratory Questions

- Response Latency | Will learners in the IMPASSE condition will spend more time on the first question than learners in CONTROL?

- Consistency | How consistent are learners in their interpretation of the graph? Do they adopt an interpretation on the first question and hold constant? Or do they change interpretations from question to question? Are there any interpretations that serve as 'absorbing states' (i.e. once encountered, the learner does not exist this state).

- Time Course of Exploration | What is the relationship between response accuracy (and interpretation) and time spent on each item?
- Can exploration strategies be derived from mouse cursor activity?

1.1 METHODS

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

1.1.2 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.

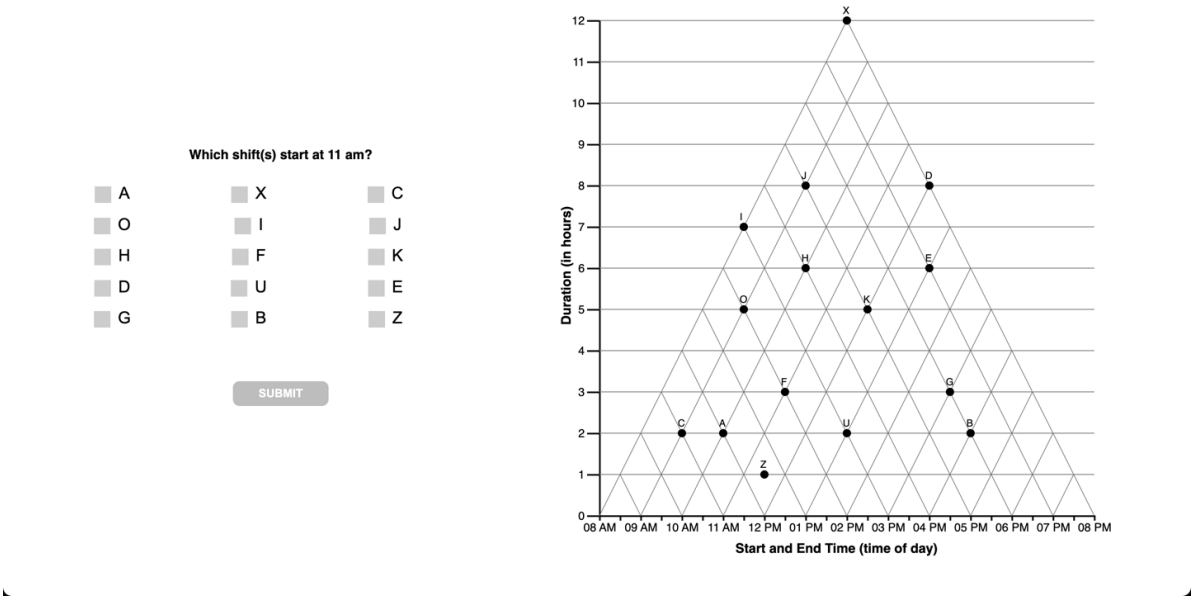


Figure 1.2: Sample Question (Q=1) for Graph Comprehension Task

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. We hypothesize that

this presents the reader with an obstacle, at which point they are forced to confront their interpretation of the coordinate system and (ideally) develop a new strategy.

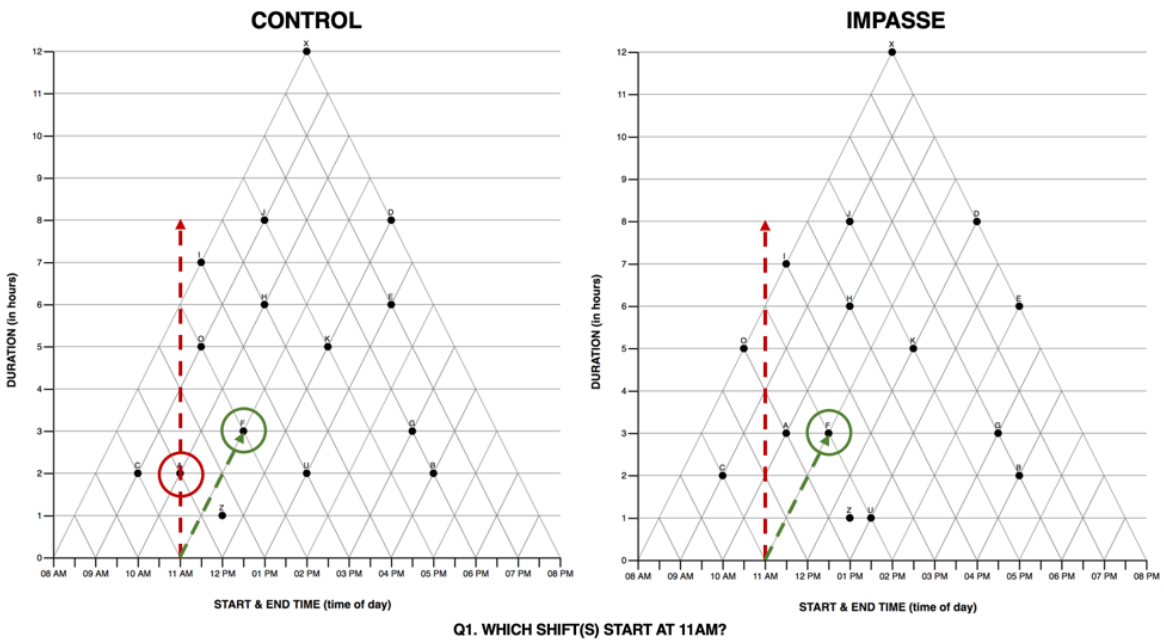


Figure 1.3: Sample Question (Q=1) graphs for each condition

1.1.3 Procedure

Participants completed the study via a web-browser.

(1) Upon starting, they submitted informed consent, before reading task instructions.

(2) 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.

(3) Then participants completed an experimental block of 15 items.

(3A) The first five items in the task are defined as the SCAFFOLDING block. 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.

(3B) The remaining 10 items are defined as the TESTING block. In both conditions, these questions were not structured as impasse (i.e. contained an available orthogonal answer)

(4) Following the experimental block, participants answered a free-response question about their strategy for reading the graph, followed by a demographic questionnaire and debrief.

1.1.4 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).

1.2 ANALYSIS

1.2.1 Data Preparation

Data were collected via a custom web application and stored in a NoSQL database. The following exclusion criteria were applied during data cleaning:

- completion status : "success" ; subject must have finished all parts of the study, including demographic questionnaire
- session ID: [in list] ; subject must have been assigned to valid data collection session (discard testing and piloting data)
- browser interaction violations < 3; subject must have fewer than 3 violations of non-allowed browser interactions (i.e. resizing window, leaving browser tab or leaving fullscreen mode)
- self-rated effort > 2; subjects who reported, "not trying hard/rushing through questions" or "started out trying hard but giving up at some point" were excluded from analysis.
- attention check ==TRUE ; subjects who failed to answer a mid-study attention check question (Graph Comprehension Task Question #6) are excluded

Before analysis, data files from individual data collection periods are harmonized into a common data format.

Pre-Requisite	Followed By
spring17_clean_data.Rmd spring18_clean_data.Rmd fall21_clean_data.Rmd winter2022_clean_sgc3a.Rmd	2_sgc3A_scoring.qmd

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).

1.2.1.1 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. No raw data (responses and response times) are discarded, only algorithmically-derived scores for the responses.

```
#IMPORT PARTICIPANT DATA

# HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# # mbp = "/Users/amyfox/Sites/RESEARCH/SGC-Scaffolding Graph Comprehension/SGC-X/ANALYSIS
# setwd(imac)

#set datafiles
fall17 <- "analysis/SGC3A/data/0-session-level/fall17_sgc3a_participants.csv"
spring18 <- "analysis/SGC3A/data/0-session-level/spring18_sgc3a_participants.csv"
fall21 <- "analysis/SGC3A/data/0-session-level/fall21_sgc3a_participants.csv"
winter22 <- "analysis/SGC3A/data/0-session-level/winter22_sgc3a_participants.rds"
```

```r
#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') %>%
  dplyr::select(
  subject,condition,term,mode,
  gender,age,language, schoolyear, country,
  effort,difficulty,confidence,enjoyment,other,
  totaltime_m,
  # absolute_score, #drop absolute score as this is re-scored [though should be the same]
  #exploratory factors
  violations, browser, width, height
)

#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",
    violations = "NULL",
    browser = "NULL",
    width = "NULL",
    height = "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, #drop absolute score as this is re-scored [though should
                #exploratory factors
                violations, browser, width, height)

#save 'explanation' columns from winter22, which is actually a response to a free response
df_winter22_q16 <- df_subjects_winter22 %>%
  dplyr::select(subject, condition, term , mode, explanation) %>%
  mutate(
    q = 16,
    response = explanation
  ) %>% dplyr::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, #drop absolute score as this is re-scored [though shoul
                 #exploratory factors
                 violations, browser, width, height)

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),
    pretty_condition = recode_factor(condition, "111" = "control", "121" =  "impasse"),
    pretty_mode = recode_factor(mode, "lab-synch" = "laboratory", "asynch" =  "online-repl
    term = factor(term, levels= c("fall17","spring18","fall21","winter22")),
    mode = factor(mode, levels=c("lab-synch","asynch")),
    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)
```

1.2.1.2 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.

```
# HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
```

```
# #mbp = "/Users/amyfox/Sites/RESEARCH/SGC-Scaffolding Graph Comprehension/SGC-X/ANALYSIS/
# setwd(imac)

#set datafiles
fall17 <- "analysis/SGC3A/data/0-session-level/fall17_sgc3a_blocks.csv"
spring18 <- "analysis/SGC3A/data/0-session-level/spring18_sgc3a_blocks.csv"
fall21 <- "analysis/SGC3A/data/0-session-level/fall21_sgc3a_blocks.csv"
winter22 <- "analysis/SGC3A/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)
  )
```

```r
#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, levels= c("fall17","spring18","fall21","winter22")),
    mode = factor(mode, levels=c("lab-synch","asynch")),
    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)
  ) %>%
  # handle NA values (why are some empty responses blank and others NA?)
  mutate(
    response = replace_na(response, ""),
    num_o = replace_na(num_o, 0)
  )


#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)

#add back pretty condition
df_items <- df_items %>% mutate(
  pretty_condition = recode_factor(condition, "111" = "control", "121" =  "impasse"),
  pretty_mode = recode_factor(mode, "lab-synch" = "laboratory", "asynch" =  "online-replic
)

#CLEANUP
rm(df_items_fall17,df_items_fall21, df_items_spring18, df_items_winter22, df_items_before,
rm(fall17,fall21,spring18,winter22, map_relations)
```

1.2.1.3 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
```

1.2.1.4 Export

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

```
# HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
```

```
# #mbp = "/Users/amyfox/Sites/RESEARCH/SGC-Scaffolding Graph Comprehension/SGC-X/ANALYSIS/
# setwd(imac)

#SAVE FILES
write.csv(df_subjects,"analysis/SGC3A/data/1-study-level/sgc3a_participants.csv", row.name
write.csv(df_items,"analysis/SGC3A/data/1-study-level/sgc3a_items.csv", row.names = FALSE)
write.csv(df_freeresponse,"analysis/SGC3A/data/1-study-level/sgc3a_freeresponse.csv", row.

#SAVE R Data Structures
#export R DATA STRUCTURES (include codebook metadata)
rio::export(df_subjects, "analysis/SGC3A/data/1-study-level/sgc3a_participants.rds") # to
rio::export(df_items, "analysis/SGC3A/data/1-study-level/sgc3a_items.rds") # to R data str
```

1.3 RESOURCES

```
sessionInfo()
```

```
R version 4.2.1 (2022-06-23)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur ... 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.2/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] kableExtra_1.3.4 forcats_0.5.1    stringr_1.4.0    dplyr_1.0.9
 [5] purrr_0.3.4      readr_2.1.2      tidyr_1.2.0      tibble_3.1.7
 [9] ggplot2_3.3.6    tidyverse_1.3.1  codebook_0.9.2

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.8.3      svglite_2.1.0     lubridate_1.8.0   assertthat_0.2.1
 [5] digest_0.6.29     utf8_1.2.2        R6_2.5.1          cellranger_1.1.0
```

```
 [9] backports_1.4.1   reprex_2.0.1      labelled_2.9.1    evaluate_0.15
[13] httr_1.4.3        pillar_1.7.0      rlang_1.0.3       curl_4.3.2
[17] readxl_1.4.0      data.table_1.14.2 rstudioapi_0.13   rmarkdown_2.14
[21] webshot_0.5.3     foreign_0.8-82    bit_4.0.4         munsell_0.5.0
[25] broom_0.8.0       compiler_4.2.1    modelr_0.1.8      xfun_0.31
[29] pkgconfig_2.0.3   systemfonts_1.0.4 htmltools_0.5.2   tidyselect_1.1.2
[33] rio_0.5.29        codetools_0.2-18  fansi_1.0.3       viridisLite_0.4.0
[37] crayon_1.5.1      tzdb_0.3.0        dbplyr_2.2.1      withr_2.5.0
[41] grid_4.2.1        jsonlite_1.8.0    gtable_0.3.0      lifecycle_1.0.1
[45] DBI_1.1.3         magrittr_2.0.3    scales_1.2.0      zip_2.2.0
[49] cli_3.3.0         stringi_1.7.6     vroom_1.5.7       fs_1.5.2
[53] xml2_1.3.3        ellipsis_0.3.2    generics_0.1.2    vctrs_0.4.1
[57] openxlsx_4.2.5    tools_4.2.1       bit64_4.0.5       glue_1.6.2
[61] hms_1.1.1         parallel_4.2.1    fastmap_1.1.0     yaml_2.3.5
[65] colorspace_2.0-3  rvest_1.0.2       knitr_1.39        haven_2.5.0
```

2 Response Scoring

TODO

- finish item level response exploration
- TODO: generate heat maps of Q9. Same answer but very different optimal operation paths!
- see individual item level todos on response exploration

The purpose of this notebook is to score (assign a measure of accuracy) to response data for the SGC3A 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.) To review the strategy behind Multiple Response scoring for the SGC project, refer to section **?@sec-scoring**.

Pre-Requisite
1_sgc3A_harmonize.qmd

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

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

2.1 SCORE SGC DATA

To review the strategy behind Multiple Response scoring for the SGC project, refer to section **?@sec-scoring**.

In SGC 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:

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.

Based on prior observational studies where we observed emergence of other alternative interpretations (i.e. transitional interpretations) we also calculate subscores for these alternatives.

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 strategy) answer key.

2.1.1 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.

```
# #HACK WD FOR LOCAL RUNNING?
#imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
#setwd(imac)

#SAVE KEYS FOR FUTURE USE
keys_raw <-  read_csv("analysis/utils/keys/parsed_keys/keys_raw")
keys_orth <-  read_csv("analysis/utils/keys/parsed_keys/keys_orth")
keys_tri <-  read_csv("analysis/utils/keys/parsed_keys/keys_tri")
keys_satisfice_left <-  read_csv("analysis/utils/keys/parsed_keys/keys_satisfice_left")
keys_satisfice_right <-  read_csv("analysis/utils/keys/parsed_keys/keys_satisfice_right")
keys_tversky_duration <-  read_csv("analysis/utils/keys/parsed_keys/keys_tversky_duration"
keys_tversky_end <-  read_csv("analysis/utils/keys/parsed_keys/keys_tversky_end")
keys_tversky_max <-  read_csv("analysis/utils/keys/parsed_keys/keys_tversky_max")
keys_tversky_start <-  read_csv("analysis/utils/keys/parsed_keys/keys_tversky_start")
```

2.1.2 Calculate Subscores

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** interpretation?
  - `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). This is a more generous version of the Absolute score that does not penalize the participant if in addition to the correct answer *in addition to* they also select the data point referenced in the question.

```
#HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# setwd(imac)

#backup <- read_rds('analysis/SGC3A/data/1-study-level/sgc3a_items.rds') #for troubleshoot
df_items <- read_rds('analysis/SGC3A/data/1-study-level/sgc3a_items.rds')
```

```
# #HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# setwd(imac)

source("analysis/utils/scoring.R")
```

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 PARTIAL-SUBSCORES
df_items$score_TRI = pbmapply(calc_subscore, df_items$q, df_items$condition, df_items$resp
df_items$score_ORTH = pbmapply(calc_subscore, df_items$q, df_items$condition, df_items$res
df_items$score_SAT_left = pbmapply(calc_subscore, df_items$q, df_items$condition, df_items
df_items$score_SAT_right = pbmapply(calc_subscore, df_items$q, df_items$condition, df_item
df_items$score_TV_max = pbmapply(calc_subscore, df_items$q, df_items$condition, df_items$r
df_items$score_TV_start = pbmapply(calc_subscore, df_items$q, df_items$condition, df_items
df_items$score_TV_end = pbmapply(calc_subscore, df_items$q, df_items$condition, df_items$r
df_items$score_TV_duration = pbmapply(calc_subscore, df_items$q, df_items$condition, df_it

#SPECIAL ABSOLUTE SUBSCORES
df_items$score_REF = pbmapply(calc_refscore, df_items$q, df_items$response)
df_items$score_BOTH = as.integer((df_items$score_TRI == 1) & (df_items$score_ORTH ==1))

#ABSOLUTE SCORES
df_items$score_ABS = as.integer(df_items$correct)
df_items$score_niceABS  <- as.integer((df_items$score_TRI == 1)) #tri doesn't penalize ref
```

2.1.3 Derive Interpretation

Finally, we use the interpretation subscores to classify the response as a particular interpretation. This classification algorithm : (1) First decides if the response matches one or more 'special' situations (blank response, reference point response, both ORTH and TRI) (2) If response doesn't match a special situation, it compares the individual subscores, and subscores and decides if they are *discriminant* (i.e. are the scores different enough to make a prediction). A discriminant threshold of 0.5pts (on a scale from -1 to +1 is used) (2) If the variance in subscores surpasses the threshold, the interpretation is classified based on the highest subscore (TRIANGULAR, ORTHOGONAL, TVERSKY, SATISFICE) (3) If the variance does not surpass the threshold, the interpretation is labelled as "?", indicating it cannot be classified, and is of an unknown interpretation.

The final output is called `interpretation`.

```
#stoopid extra copying for troubleshooting safety
temp <- df_items
temp <- derive_interpretation(temp)
df_items <- temp
```

2.1.4 Derive Scaled Score

The `interpretation` response variable gives us the finest grain indication of the reader's understanding of the graph for a particular question. However, it is a categorical variable, which poses a challenge for analyzing cumulative performance at the subject level. To address this challenge, we derive a *scaled_score* that converts each possible interpretation to a numeric value on a scale from -1 to +1. This scaling takes advantage of the observation that each interpretation can be positioned along a spectrum of understanding from completely incorrect (orthogonal) to completely correct (triangular). Alternative interpretations lay somewhere between.

Specifically, we assign the following values to each interpretation:

- (-1) : ORTHOGONAL, SATISFICE (satisfice represents an attempt at an orthogonal answer when none is available)
- (-0.5): ? (some alternative that cannot be identified; but meaningful that it is not orthogonal)
- (0): REFERENCE POINT, BLANK (indicates the individual thinks there is no answer, recognizes that ORTHOGONAL cannot be correct, but does not conceive of triangular)
- (+0.5) TVERSKY, BOTH TRI + ORTH (indicates that they "see" a triangular response, but lack certainty and also select the ORTHOGONAL response)
- (+1) TRIANGULAR +1

```
df_items$score_SCALED <- calc_scaled(df_items$interpretation)
```

2.2 SUMMARIZE BY SUBJECT

Next, we summarize the item level scores by subject in order to calculate cummulative sub-scores to be stored on the subject record.

```
# #HACK WD FOR LOCAL RUNNING?
# imac = "/Users/amyraefox/Code/SGC-Scaffolding_Graph_Comprehension/SGC-X/ANALYSIS/MAIN"
# setwd(imac)

#import subjects
df_subjects <- read_rds('analysis/SGC3A/data/1-study-level/sgc3a_participants.rds') %>% mu

#make temporary copies for testing safety
s = df_subjects
i = df_items

#summarize
test_subs <- summarise_bySubject(s,i)
```

```
`summarise()` has grouped output by 'subject'. You can override using the
`.groups` argument.
```

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

```
df_subjects <- test_subs
```

We also summarize absolute and scaled score progress at each question in the task, to explore cumulative performance over the task.

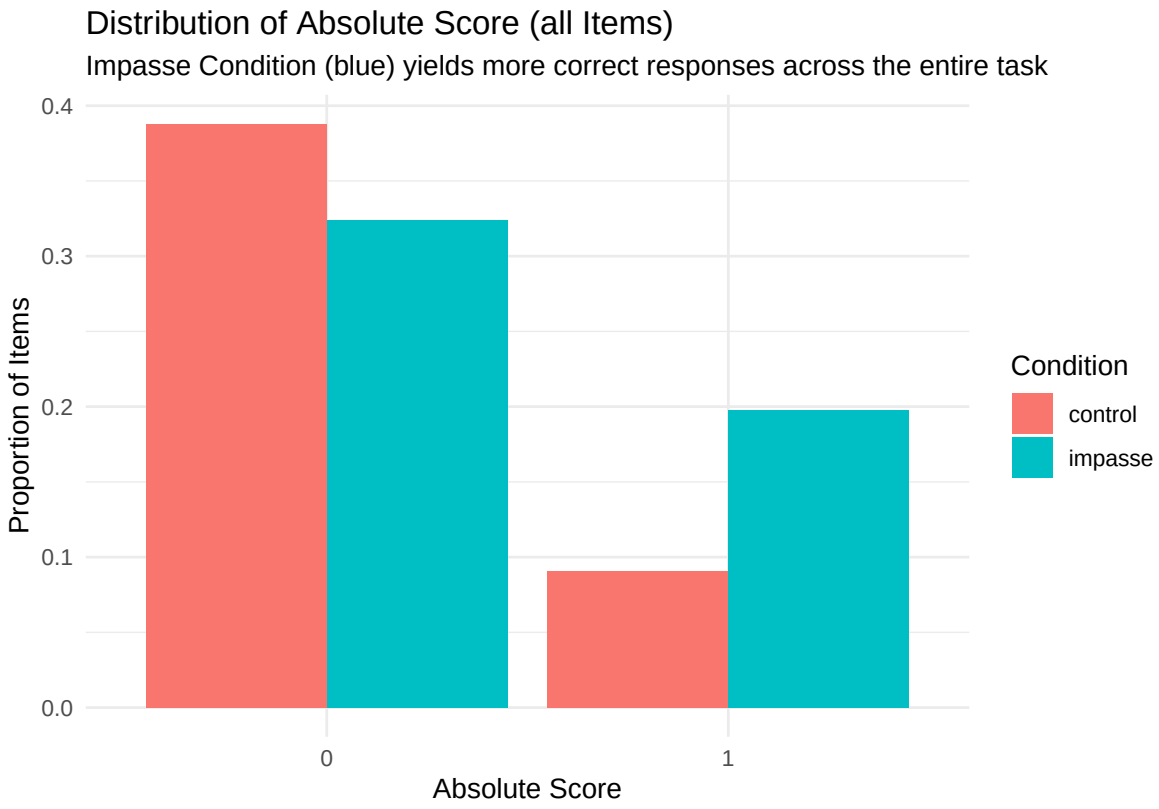
```
#GET ABSOLUTE PROGRESS
df_absolute_progress <- progress_Absolute(df_items)

#GET SCALED PROGRESS
df_scaled_progress <- progress_Scaled(df_items)
```

2.3 EXPLORE DISTRIBUTIONS

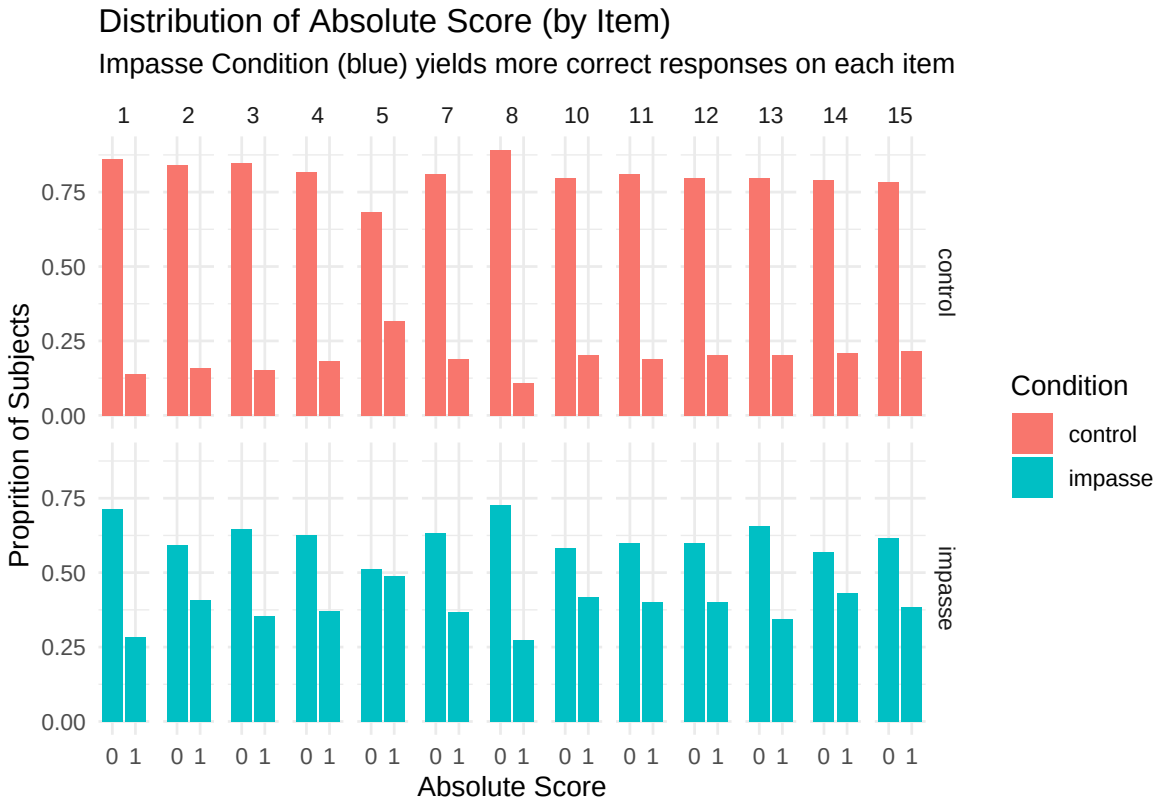
```
options(repr.plot.width =9, repr.plot.height =12)

#create temp data frame for visualizations
df = df_items %>% filter (q %nin% c(6,9)) %>% mutate(
  score_niceABS = as.factor(score_niceABS),
  pretty_interpretation = factor(interpretation,
    levels = c("Orthogonal", "Satisfice",
               "frenzy","?",
                "reference","blank",
                "Tversky", "both tri + orth",
               "Triangular" ))
  )
```

2.3.1 Absolute Score

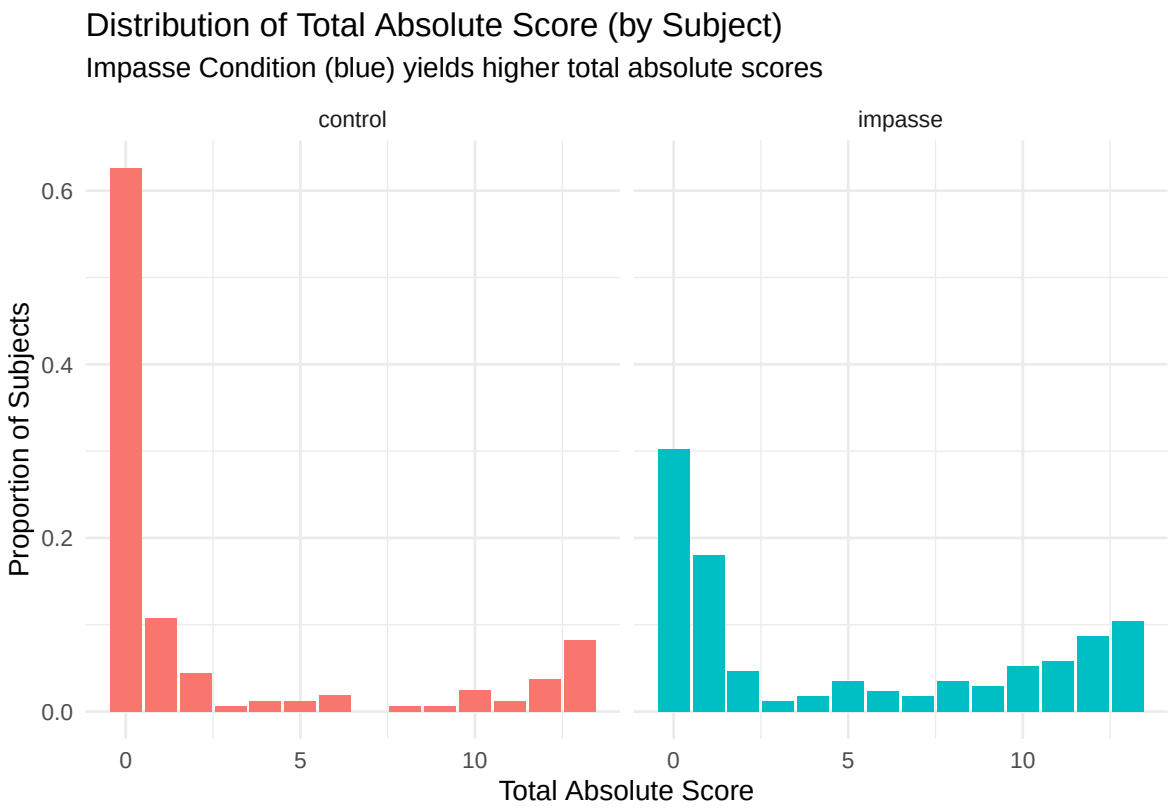
```
#DISTRIBUTION ABSOLUTE SCORE FULL TASK
gf_props(~score_niceABS, fill = ~pretty_condition, position = position_dodge(), data = df)
  labs( x = "Absolute Score",
        title = "Distribution of Absolute Score (all Items)",
        subtitle = paste("Impasse Condition (blue) yields more correct responses across th
        y = "Proportion of Items") +
  scale_fill_discrete(name = "Condition") +
  theme_minimal()
```



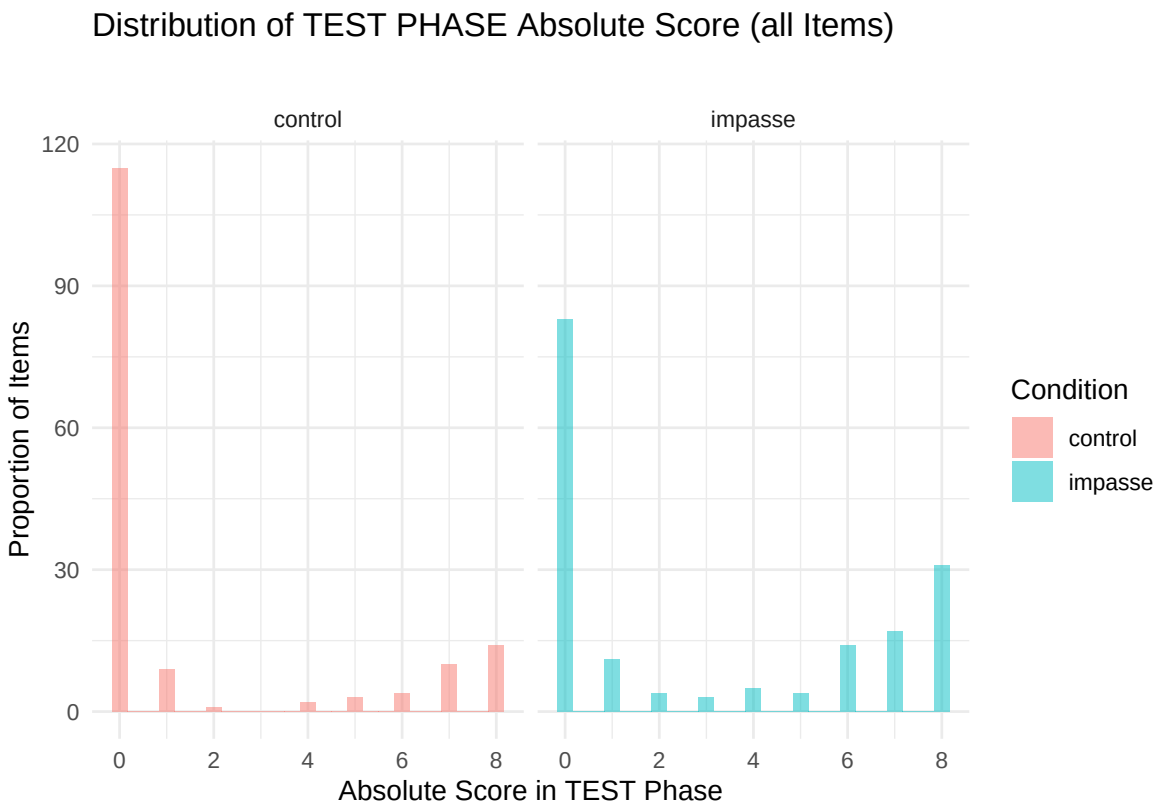

```
#DISTRIBUTION ABSOLUTE SCORE BY ITEM
gf_props(~score_niceABS, fill = ~pretty_condition, position = position_dodge(), data = df)
  gf_facet_grid(pretty_condition~q) +
  labs( x = "Absolute Score",
        title = "Distribution of Absolute Score (by Item)",
        subtitle = "Impasse Condition (blue) yields more correct responses on each item",
        y = "Proprition of Subjects") +
  scale_fill_discrete(name = "Condition") +
  theme_minimal()
```




```
#DISTRIBUTION ABSOLUTE SCORE BY SUBJECT
gf_props(~s_NABS, fill = ~pretty_condition, position = position_dodge(), data = df_subject
gf_facet_wrap(~pretty_condition)+
  labs( x = "Total Absolute Score",
        title = "Distribution of Total Absolute Score (by Subject)",
        subtitle = "Impasse Condition (blue) yields higher total absolute scores",
        y = "Proportion of Subjects") +
  scale_fill_discrete(name = "Condition") +
  theme_minimal() + theme(legend.position = "blank")
```

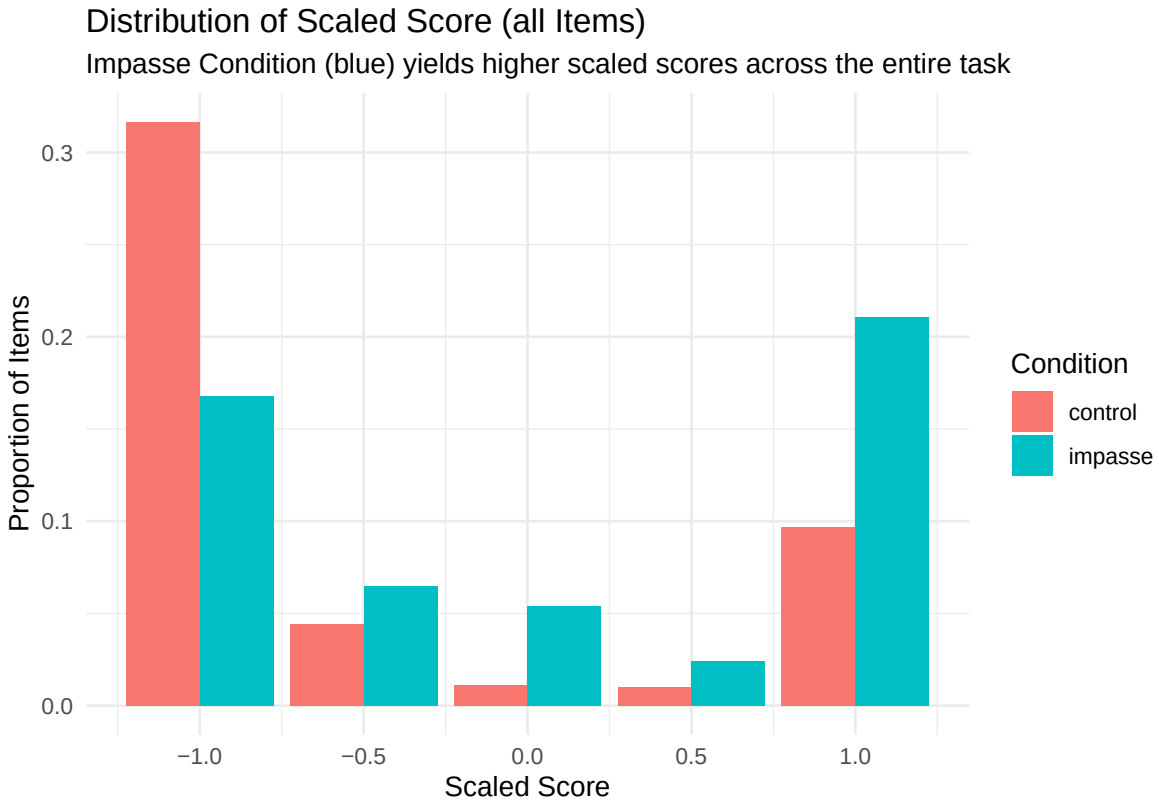



```
#DISTRIBUTION ABSOLUTE SCORE TEST PHASE
gf_histogram(~item_test_NABS, fill = ~pretty_condition,
             data = df_subjects) %>%
  gf_facet_wrap(~pretty_condition) +
  labs( x = "Absolute Score in TEST Phase",
        title = "Distribution of TEST PHASE Absolute Score (all Items)",
        subtitle = paste(""),
        y = "Proportion of Items") +
  scale_fill_discrete(name = "Condition") +
  theme_minimal()
```

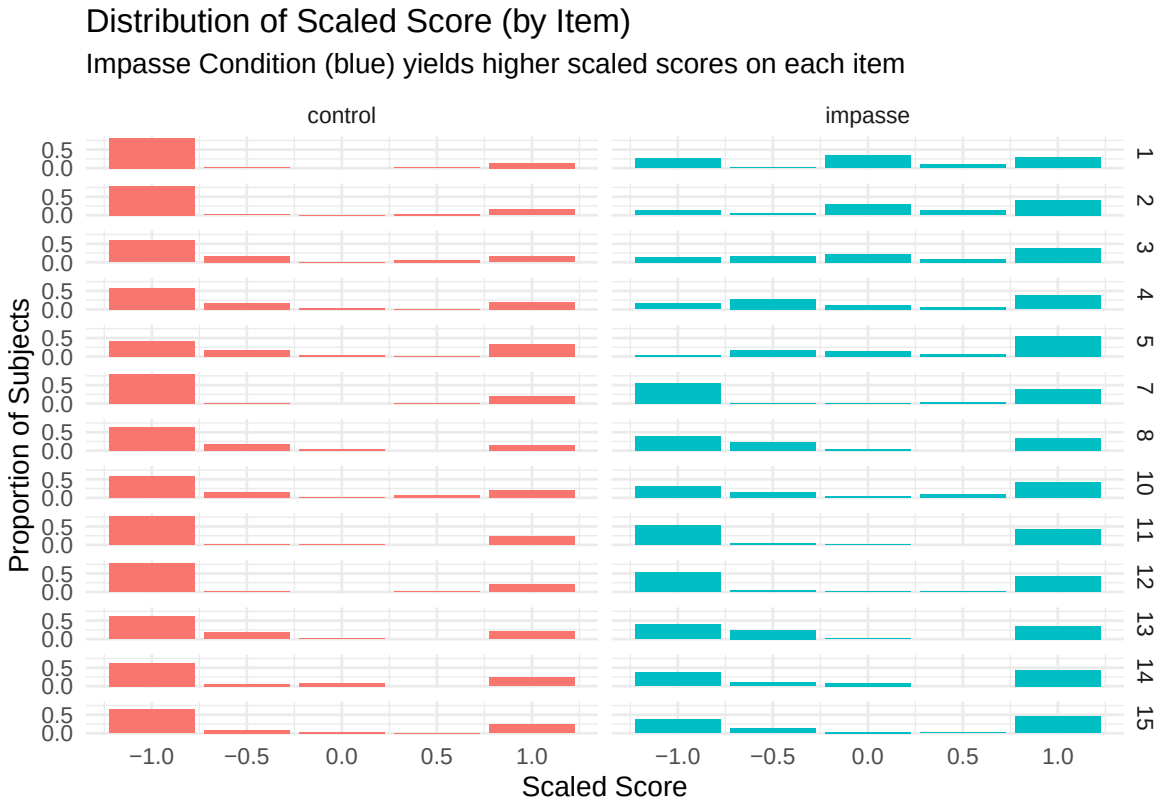


2.3.2 Scaled Score

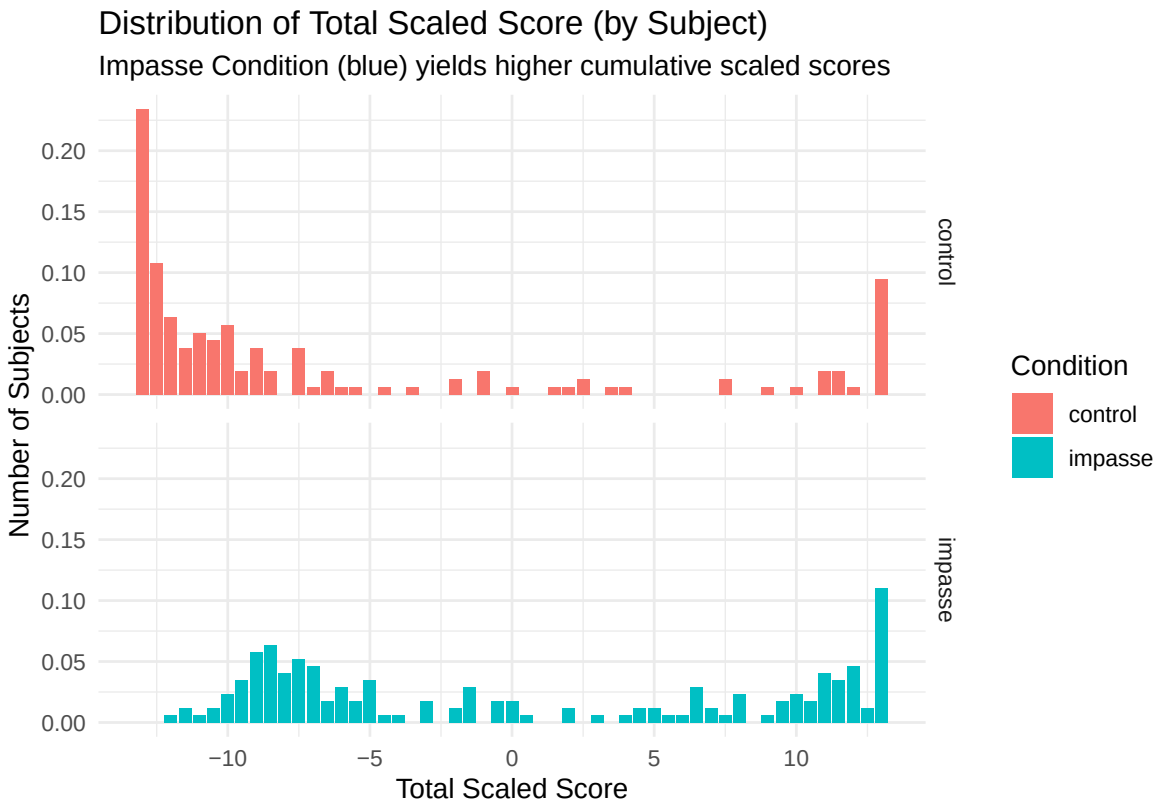
```
options(repr.plot.width =9, repr.plot.height =12)

#DISTRIBUTION SCALED SCORE FULL TASK
gf_props(~score_SCALED, fill = ~pretty_condition, position = position_dodge(), data = df)
  labs( x = "Scaled Score",
        title = "Distribution of Scaled Score (all Items)",
        subtitle = "Impasse Condition (blue) yields higher scaled scores across the entire
        y = "Proportion of Items") +
  scale_fill_discrete(name = "Condition") +
  theme_minimal()
```



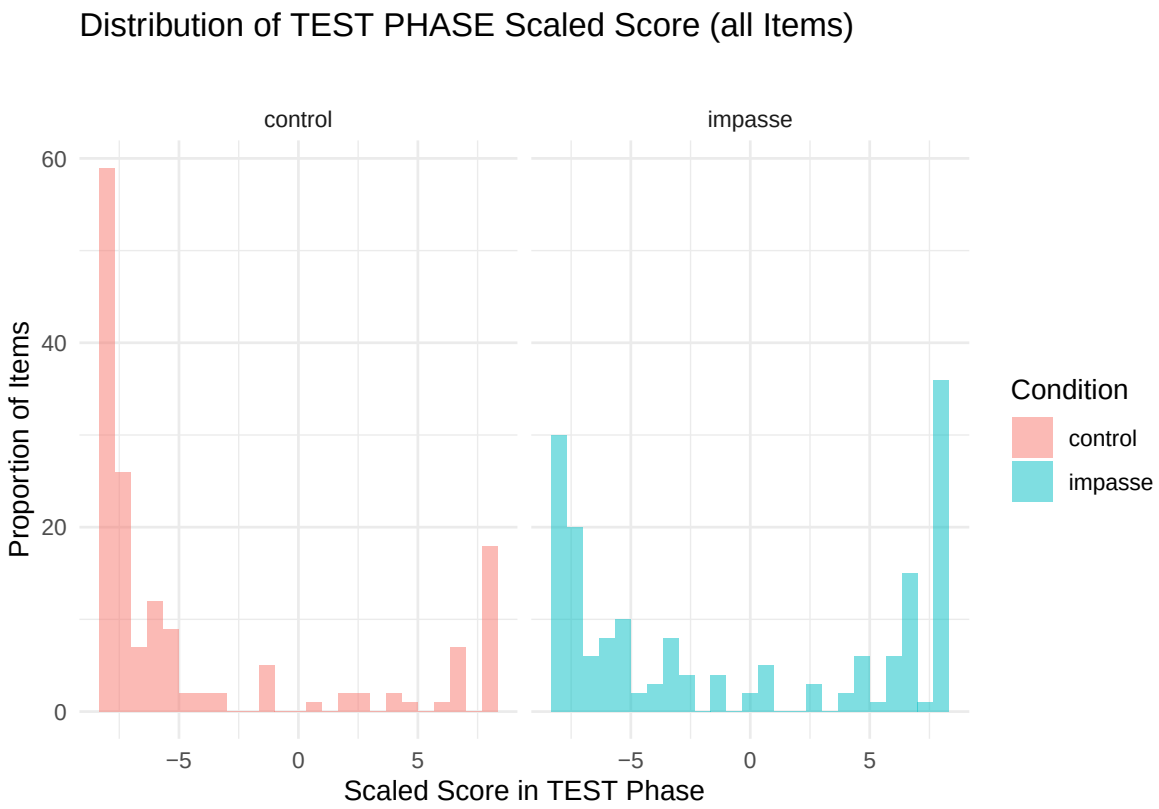

```
#DISTRIBUTION SCALED SCORE BY ITEM
gf_props(~score_SCALED, fill = ~pretty_condition, position = position_dodge(), data = df)
  gf_facet_grid(q~pretty_condition) +
  labs( x = "Scaled Score",
        title = "Distribution of Scaled Score (by Item)",
        subtitle = "Impasse Condition (blue) yields higher scaled scores on each item",
        y = "Proportion of Subjects") +
  scale_fill_discrete(name = "Condition") +  scale_y_continuous(breaks=c(0,0.5)) +
  theme_minimal() + theme(legend.position="blank")
```




```
#DISTRIBUTION SCALED SCORE BY SUBJECT
gf_props(~s_SCALED, fill = ~pretty_condition, data = df_subjects)  %>%
  gf_facet_grid(pretty_condition ~. )+
  labs( x = "Total Scaled Score",
        title = "Distribution of Total Scaled Score (by Subject)",
        subtitle = "Impasse Condition (blue) yields higher cumulative scaled scores",
        y = "Number of Subjects") +
  scale_fill_discrete(name = "Condition") +
  theme_minimal()
```



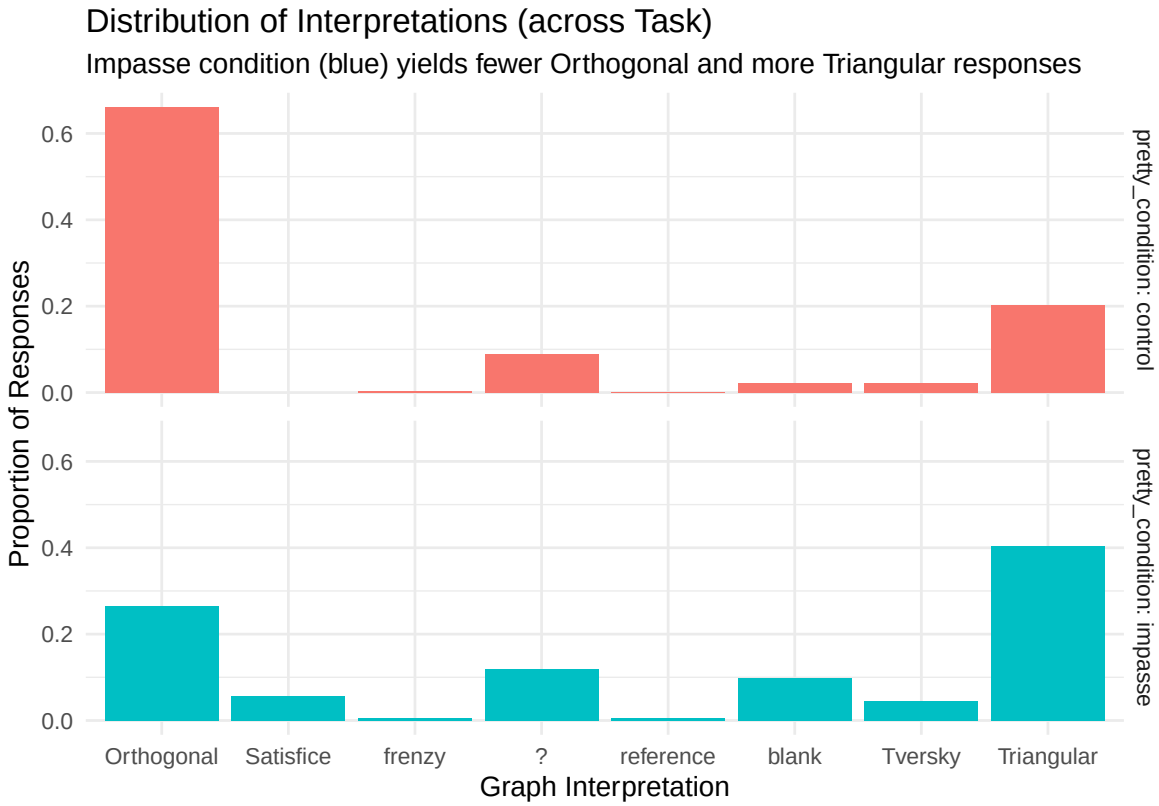

```
#DISTRIBUTION SCALED SCORE TEST PHASE
gf_histogram(~item_test_SCALED, fill = ~pretty_condition, data = df_subjects) %>%
  gf_facet_wrap(~pretty_condition) +
  labs( x = "Scaled Score in TEST Phase",
        title = "Distribution of TEST PHASE Scaled Score (all Items)",
        subtitle = paste(""),
        y = "Proportion of Items") +
  scale_fill_discrete(name = "Condition") +
  theme_minimal()
```



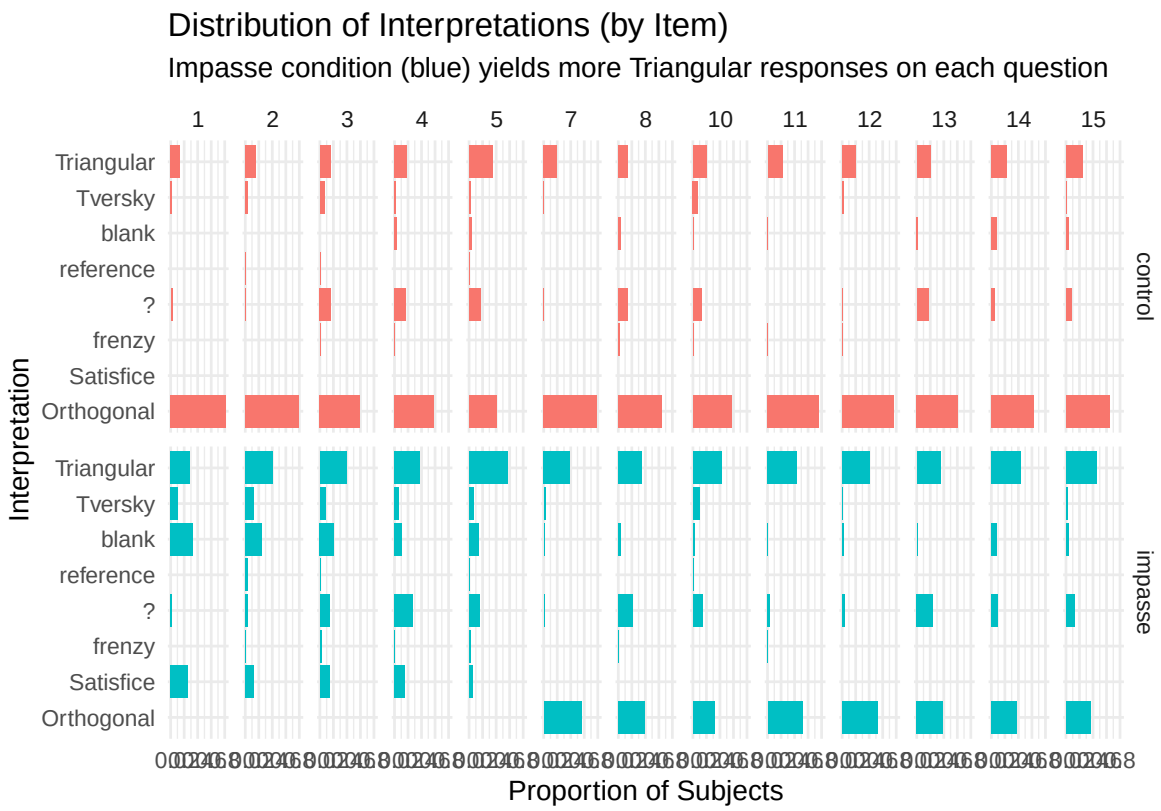
- TODO: INVESTIGATE if some of the scores assigned to 0 should be assigned to -0.5 to balance
- TODO: INVESTIGATE DISTRIBUTIONS of each subscore type

2.3.3 Interpretations

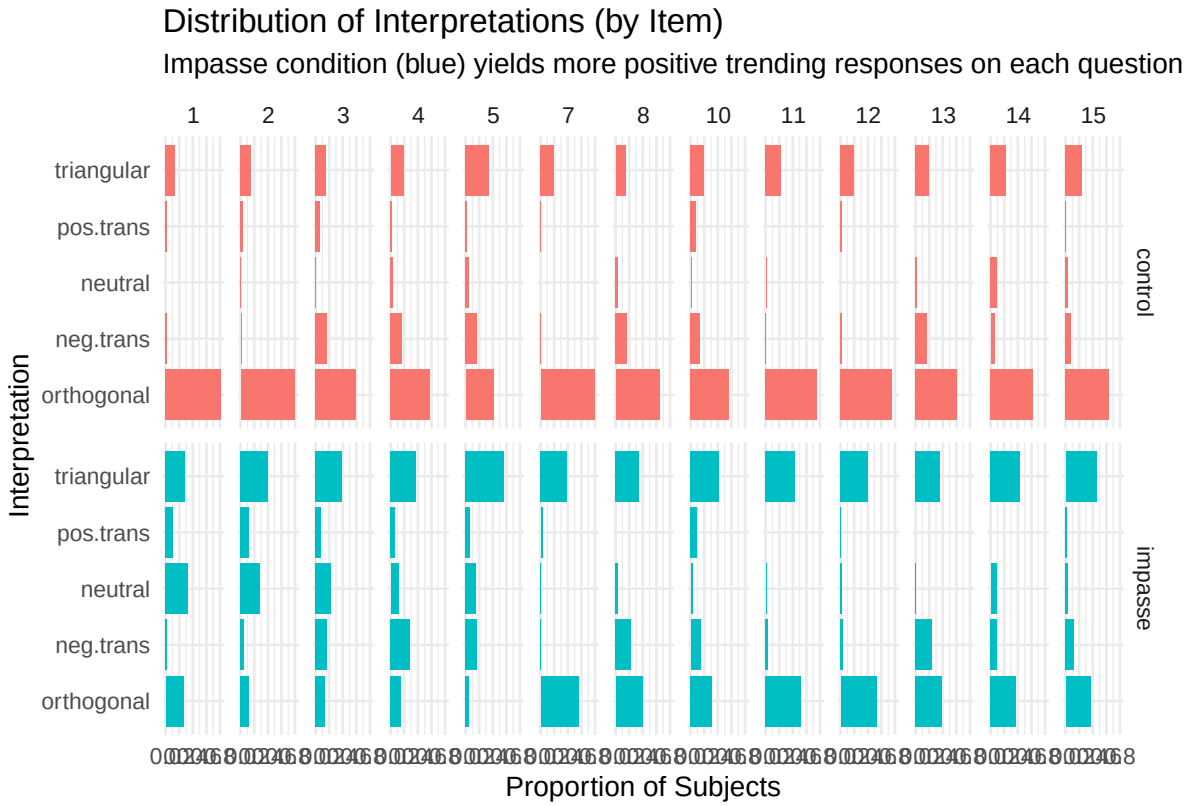
```
#DISTRIBUTION OF INTERPRETATION
gf_props(~pretty_interpretation, fill = ~pretty_condition, data = df) %>%
  gf_facet_grid( pretty_condition ~ ., labeller = label_both) +
  labs( title = "Distribution of Interpretations (across Task)",
        x = "Graph Interpretation",
        y = "Proportion of Responses",
        subtitle = "Impasse condition (blue) yields fewer Orthogonal and more Triangular r
  theme_minimal() + theme(legend.position = "blank")
```




```
#DISTRIBUTION OF INTERPRETATION ACROSS ITEMS
gf_propsh(~ pretty_interpretation, fill = ~pretty_condition, data = df) %>%
  gf_facet_grid( pretty_condition~q) +
  labs( title = "Distribution of Interpretations (by Item)",
        subtitle = "Impasse condition (blue) yields more Triangular responses on each ques
        y = "Interpretation", x = "Proportion of Subjects") + theme_minimal() + theme(lege
```

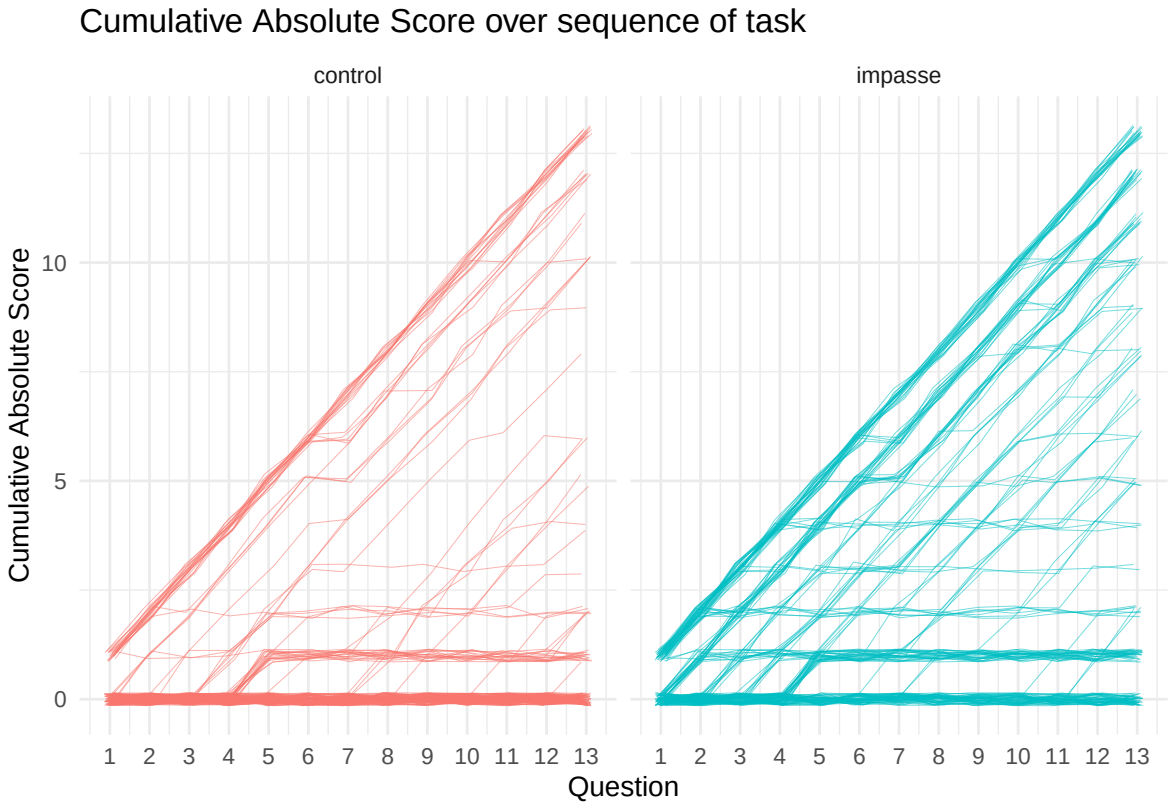



```
#DISTRIBUTION OF INTERPRETATION TYPE ACROSS ITEMS
gf_propsh(~ high_interpretation, fill = ~pretty_condition, data = df) %>%
  gf_facet_grid( pretty_condition~q) +
  labs( title = "Distribution of Interpretations (by Item)",
        subtitle = "Impasse condition (blue) yields more positive trending responses on ea
        y = "Interpretation", x = "Proportion of Subjects") + theme_minimal() + theme(lege
```

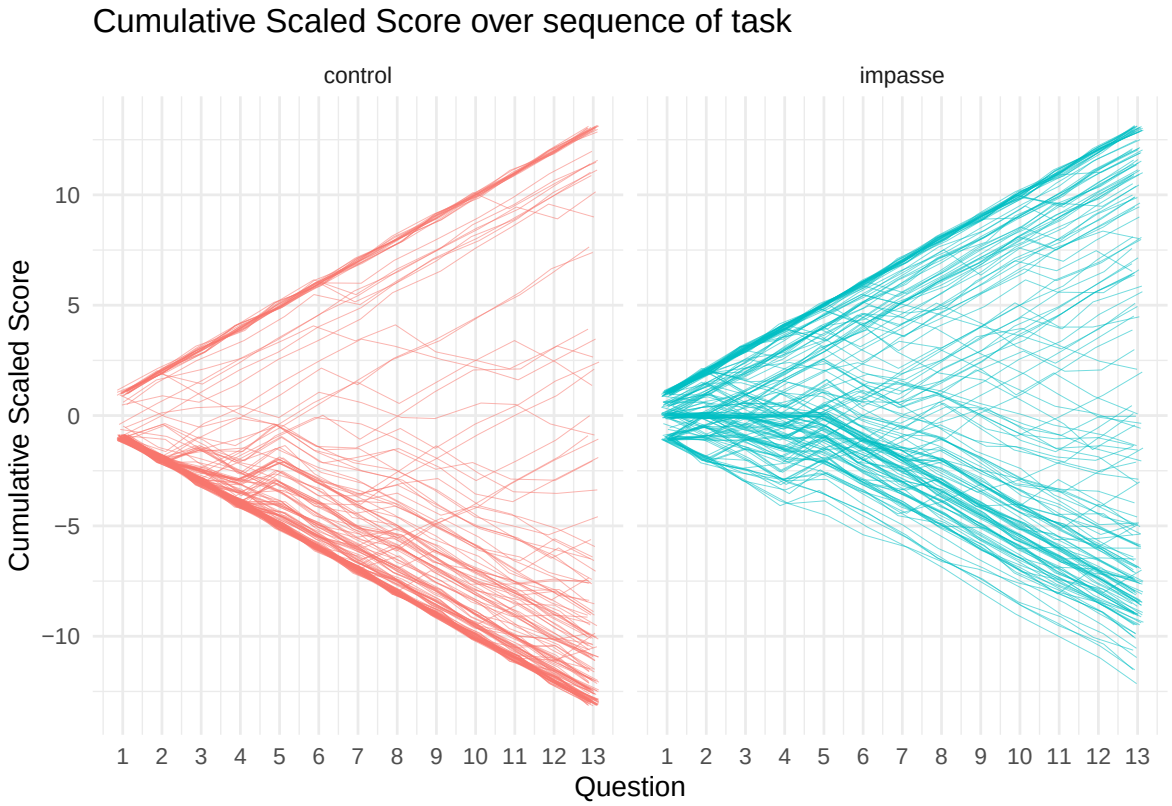


2.3.4 Progress over Task

```
#VISUALIZE progress over time ABSOLUTE score
ggplot(data = df_absolute_progress, aes(x = question, y = score, group = subject, alpha =
 geom_line(position=position_jitter(w=0.15, h=0.15), size=0.1) +
 facet_wrap(~pretty_condition) +
 labs (title = "Cumulative Absolute Score over sequence of task", x = "Question" , y = "Cu
 scale_x_continuous(breaks = c(1,2,3,4,5,6,7,8,9,10,11,12,13)) +
 theme_minimal() + theme(legend.position = "blank")
```

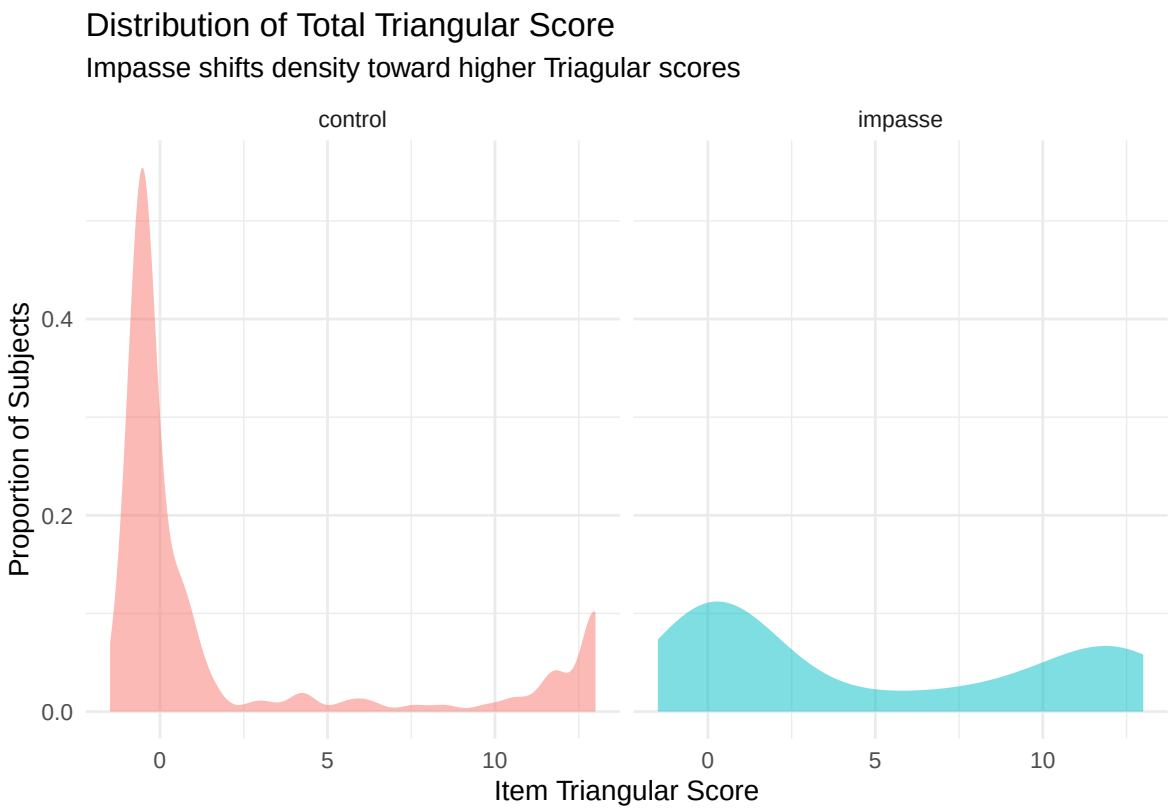



```
#VISUALIZE progress over time SCALED score
ggplot(data = df_scaled_progress, aes(x = question, y = score, group = subject, alpha = 0.
 geom_line(position=position_jitter(w=0.15, h=0.15), size=0.1) +
 facet_wrap(~pretty_condition) +
 labs (title = "Cumulative Scaled Score over sequence of task", x = "Question" , y = "Cumu
 scale_x_continuous(breaks = c(1,2,3,4,5,6,7,8,9,10,11,12,13)) +
 theme_minimal() + theme(legend.position = "blank")
```

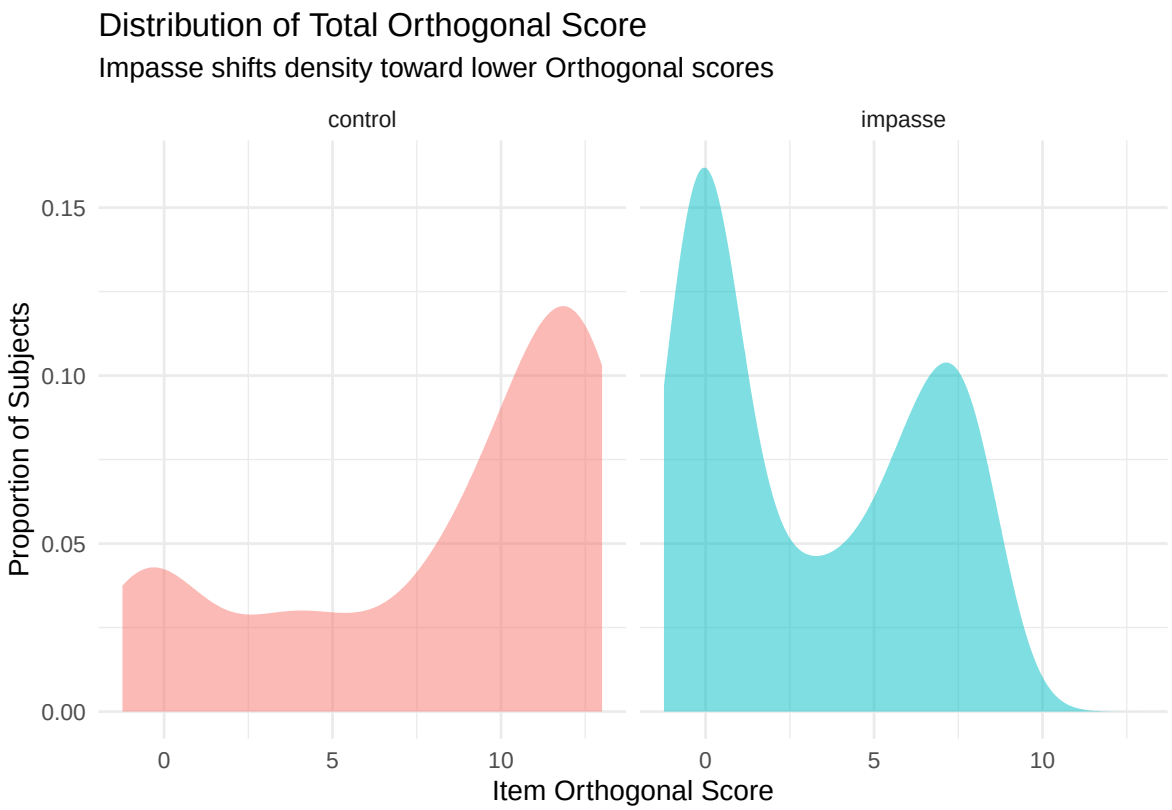


2.3.5 Interpretation Subscores

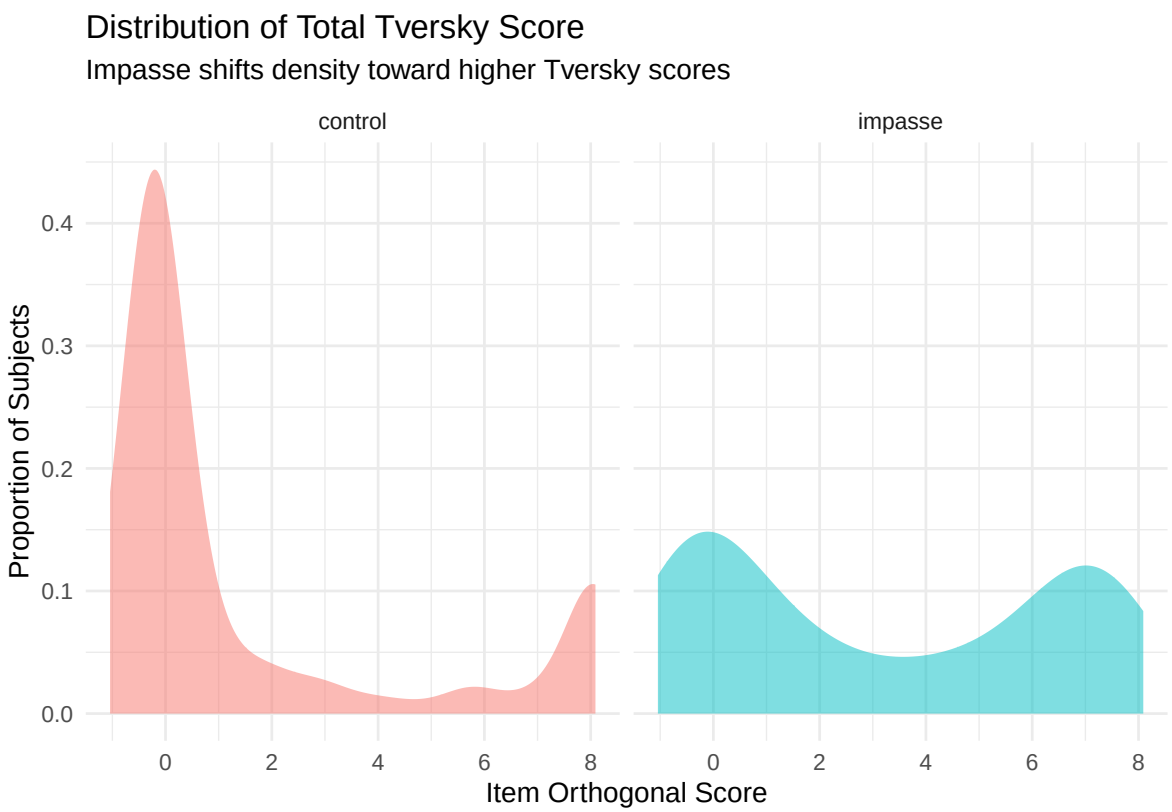
```
gf_density(~ s_TRI, fill = ~pretty_condition, data = df_subjects) %>%
  gf_facet_wrap( ~ pretty_condition) +
  labs( title = "Distribution of Total Triangular Score",
        subtitle = "Impasse shifts density toward higher Triagular scores",
        x = "Item Triangular Score", y = "Proportion of Subjects") +
        theme_minimal() + theme(legend.position = "blank")
```



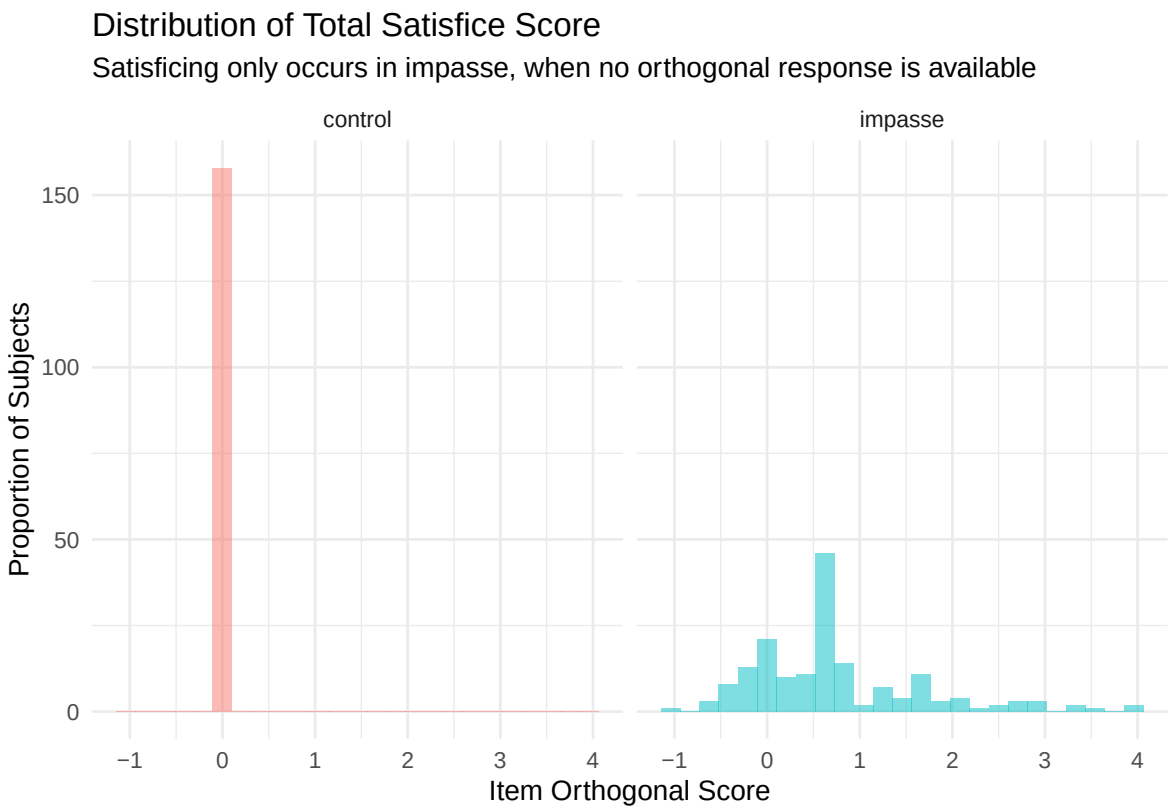

```
gf_density(~ s_ORTH, fill = ~pretty_condition, data = df_subjects) %>%
  gf_facet_wrap( ~ pretty_condition) +
  labs( title = "Distribution of Total Orthogonal Score",
        subtitle = "Impasse shifts density toward lower Orthogonal scores",
        x = "Item Orthogonal Score", y = "Proportion of Subjects") +
        theme_minimal() + theme(legend.position = "blank")
```




```
gf_density(~ s_TVERSKY, fill = ~pretty_condition, data = df_subjects) %>%
  gf_facet_wrap( ~ pretty_condition) +
  labs( title = "Distribution of Total Tversky Score",
        subtitle = "Impasse shifts density toward higher Tversky scores",
        x = "Item Orthogonal Score", y = "Proportion of Subjects") +
        theme_minimal() + theme(legend.position = "blank")
```




```
gf_histogram(~ s_SATISFICE, fill = ~pretty_condition, data = df_subjects) %>%
  gf_facet_wrap( ~ pretty_condition) +
  labs( title = "Distribution of Total Satisfice Score",
        subtitle = "Satisficing only occurs in impasse, when no orthogonal response is ava
        x = "Item Orthogonal Score", y = "Proportion of Subjects") +
        theme_minimal() + theme(legend.position = "blank")
```



2.4 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.

2.4.1 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.

2.4.1.1 Question #1

2.4.1.1.1 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).

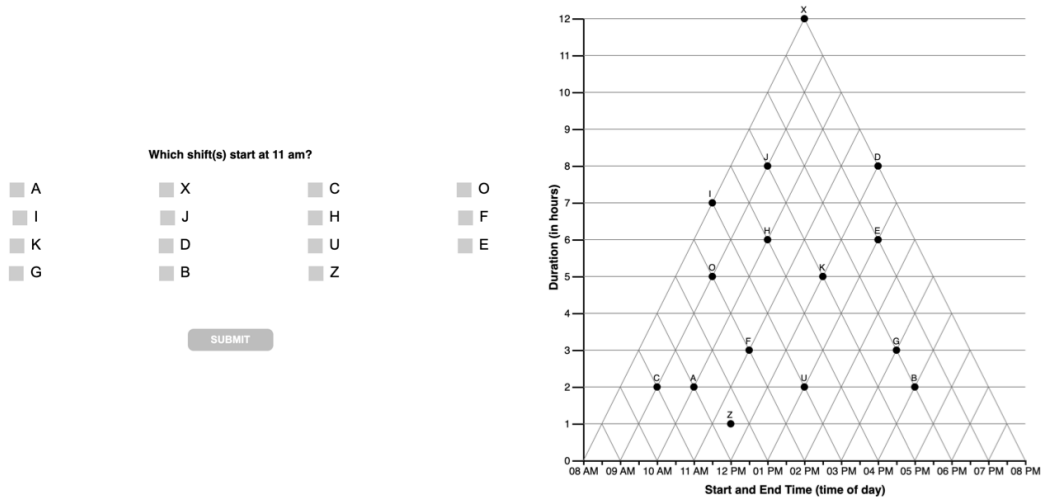


Figure 2.1: Question 1 — Control Condition

```
q <- keys_raw %>% filter(condition == "DEFAULT") %>% 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
```

Table 2.2: 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

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` indicates 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(int2),
                    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)}

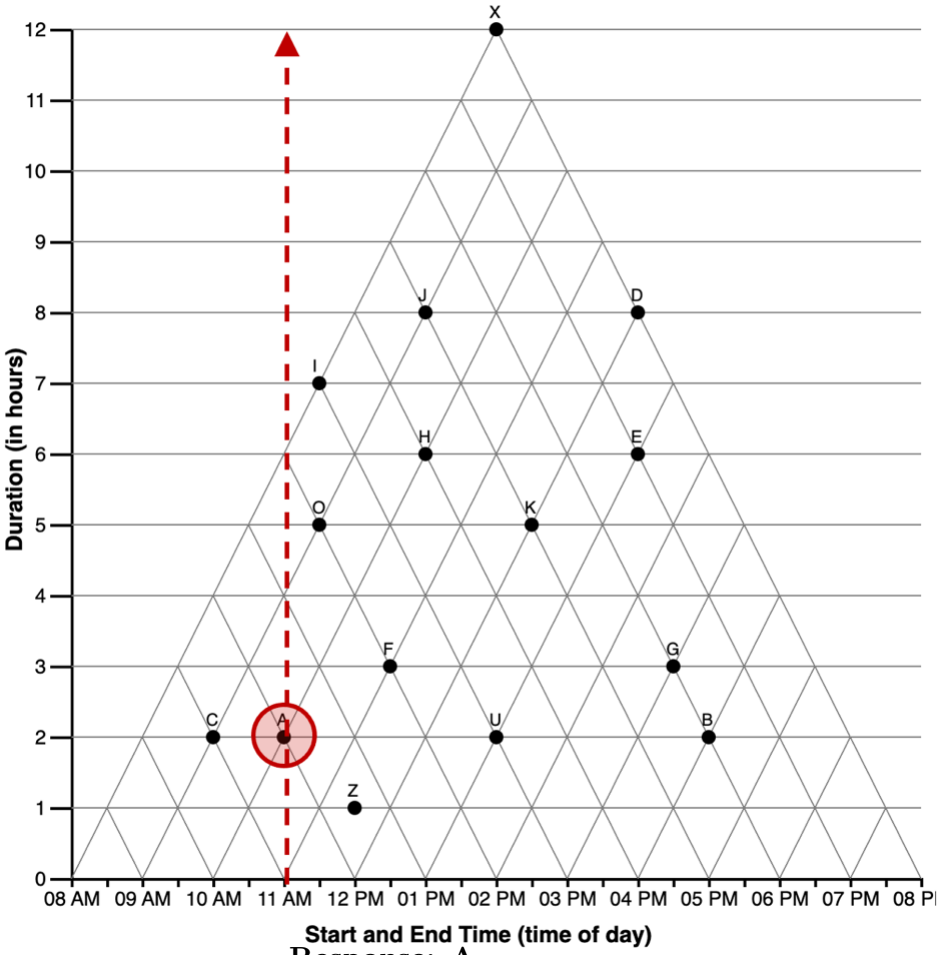
			Strict Score	Interpretation Scores				Discriminant
response	n	interpretation	absolute	tri	tversky	satisfice	orthogonal	scaled score
Triangular								
F	22	Triangular	1	1.000	1.000	NA	-0.083	1.0
Lines-Connect								
CF	3	Tversky	0	0.923	1.000	NA	-0.167	0.5
Orthogonal								
A	129	Orthogonal	0	-0.077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

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 `?` .

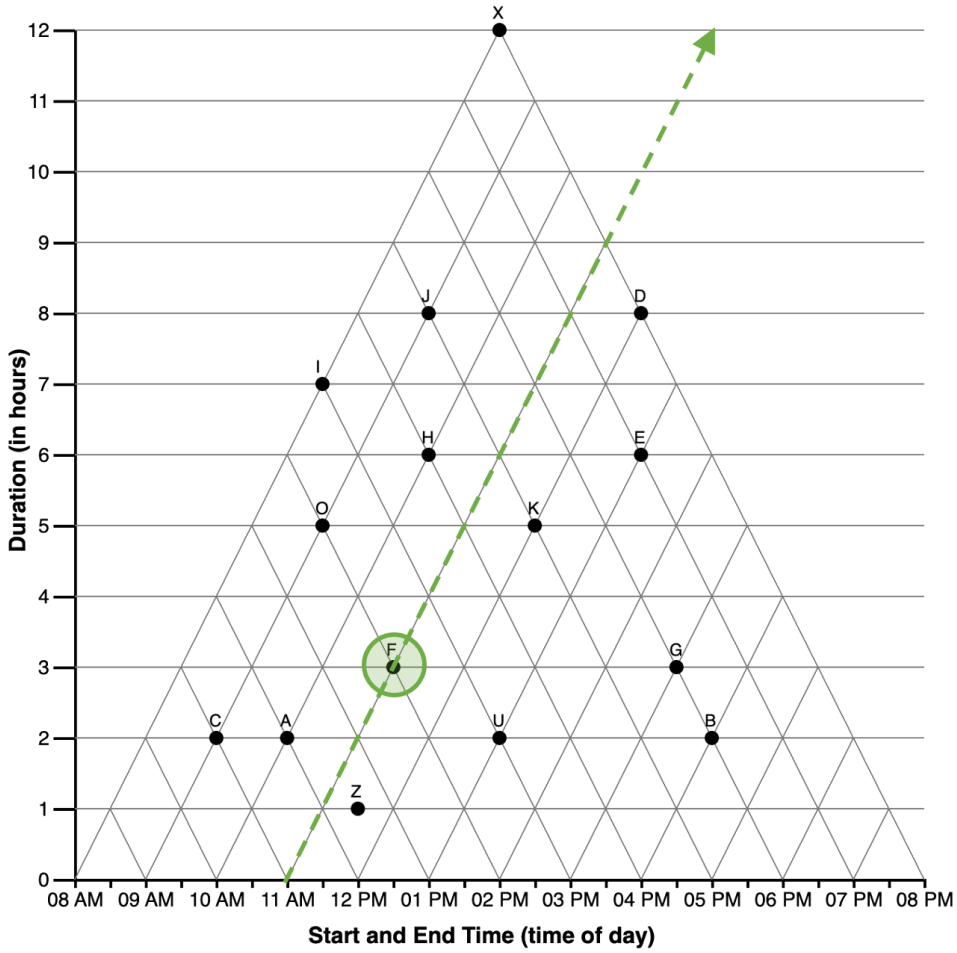
Which shifts start at
11am?



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**.

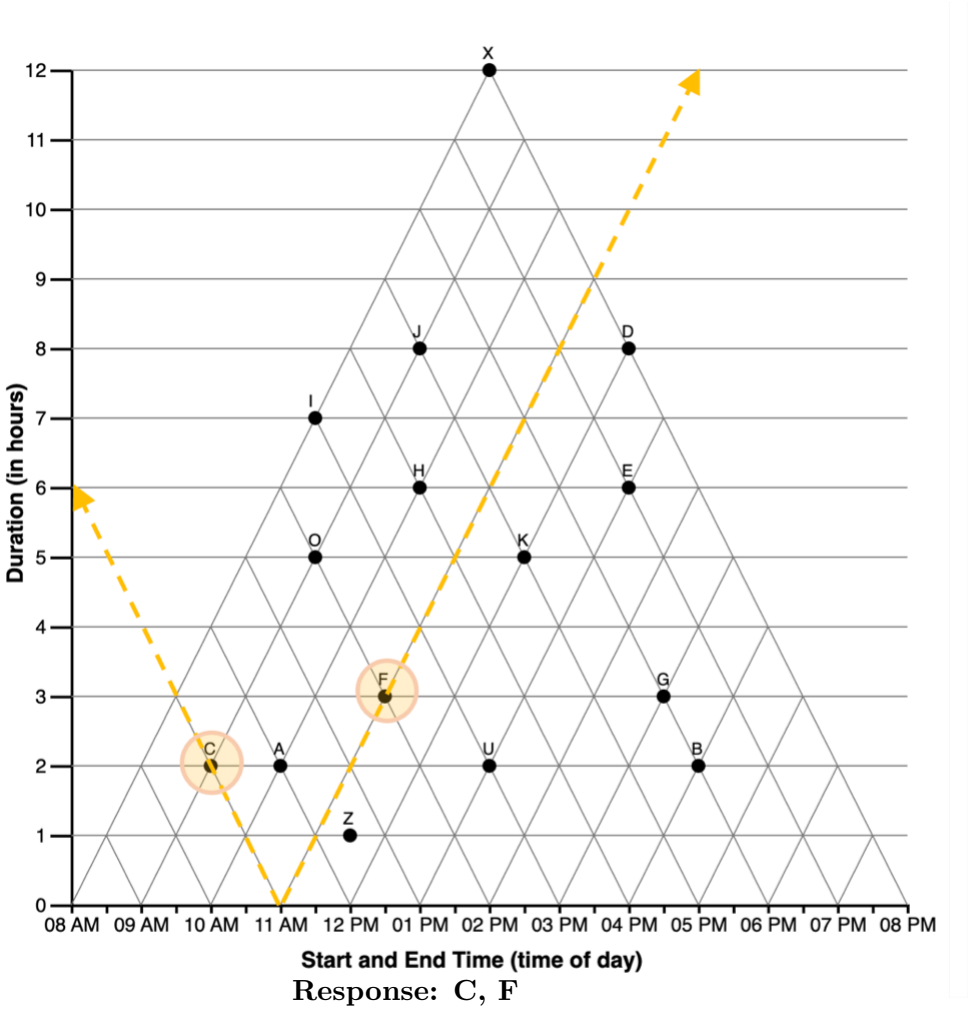
Which shifts start at
11am?



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**.

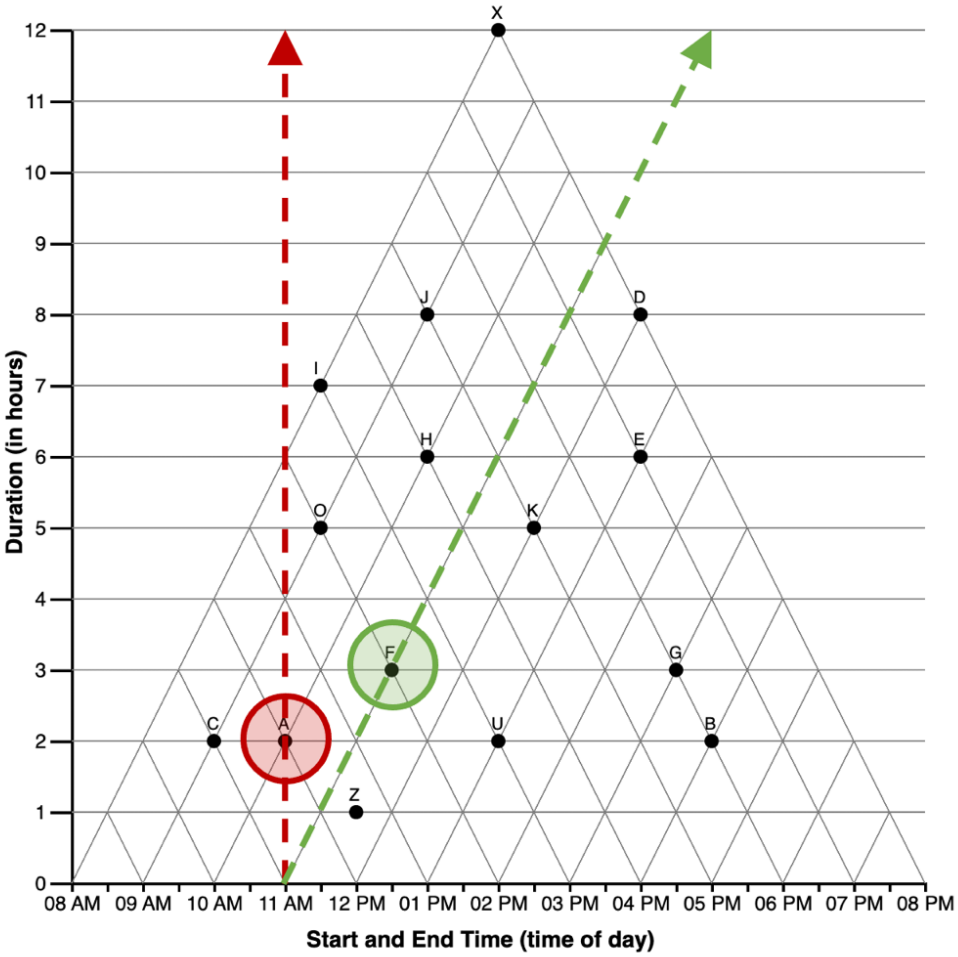
Which shifts start at
11am?



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?

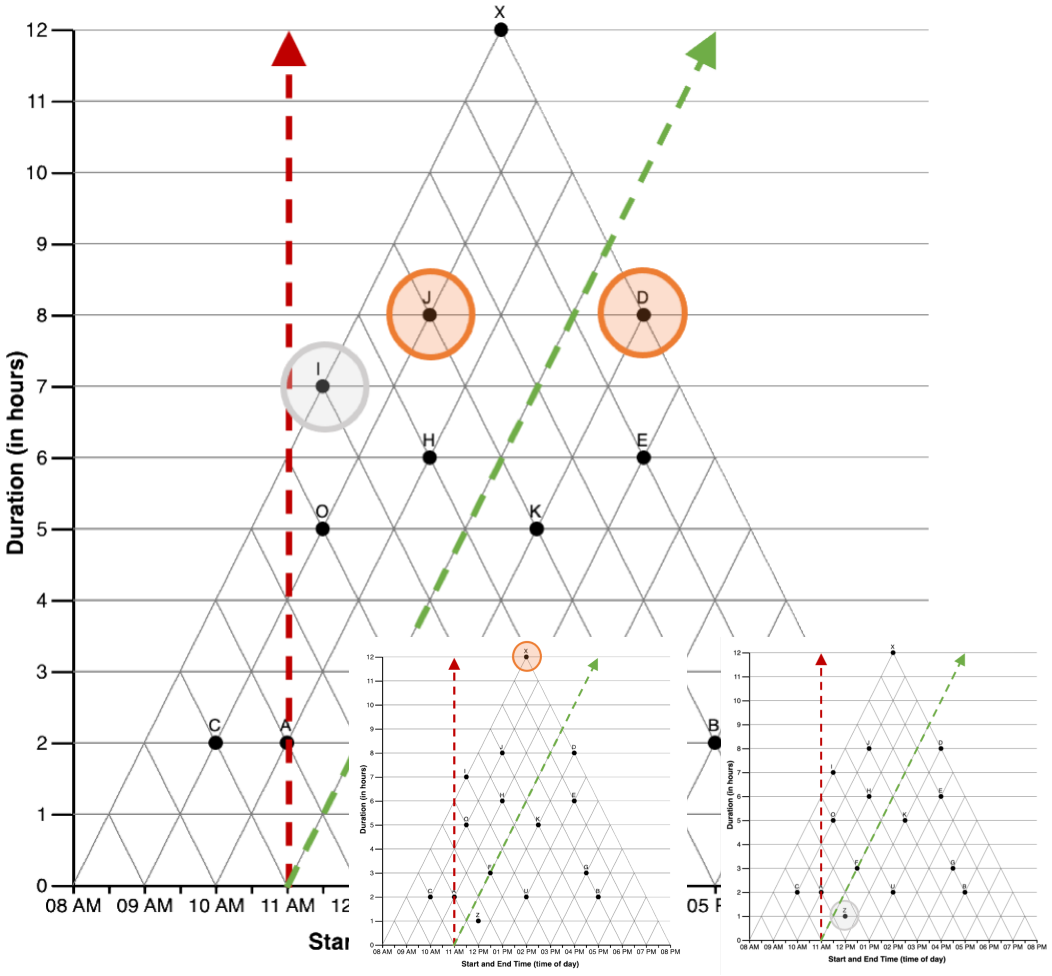


Response: A , F

- 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?



2.4.1.1.2 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
```

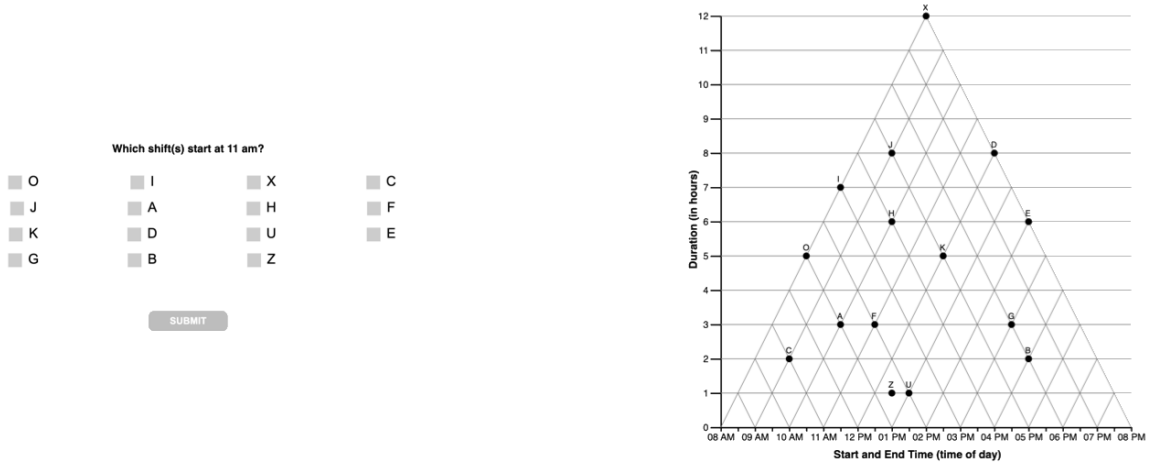


Figure 2.2: Question 1 — Impasse Condition

```
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 2.5: 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(int2),
                    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	-0.5
X	1	?	0	-0.071	-0.071	-0.071	NA	-0.5

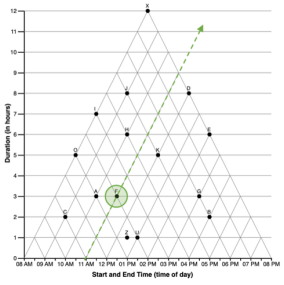
Note: n = number of responses in sample

\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

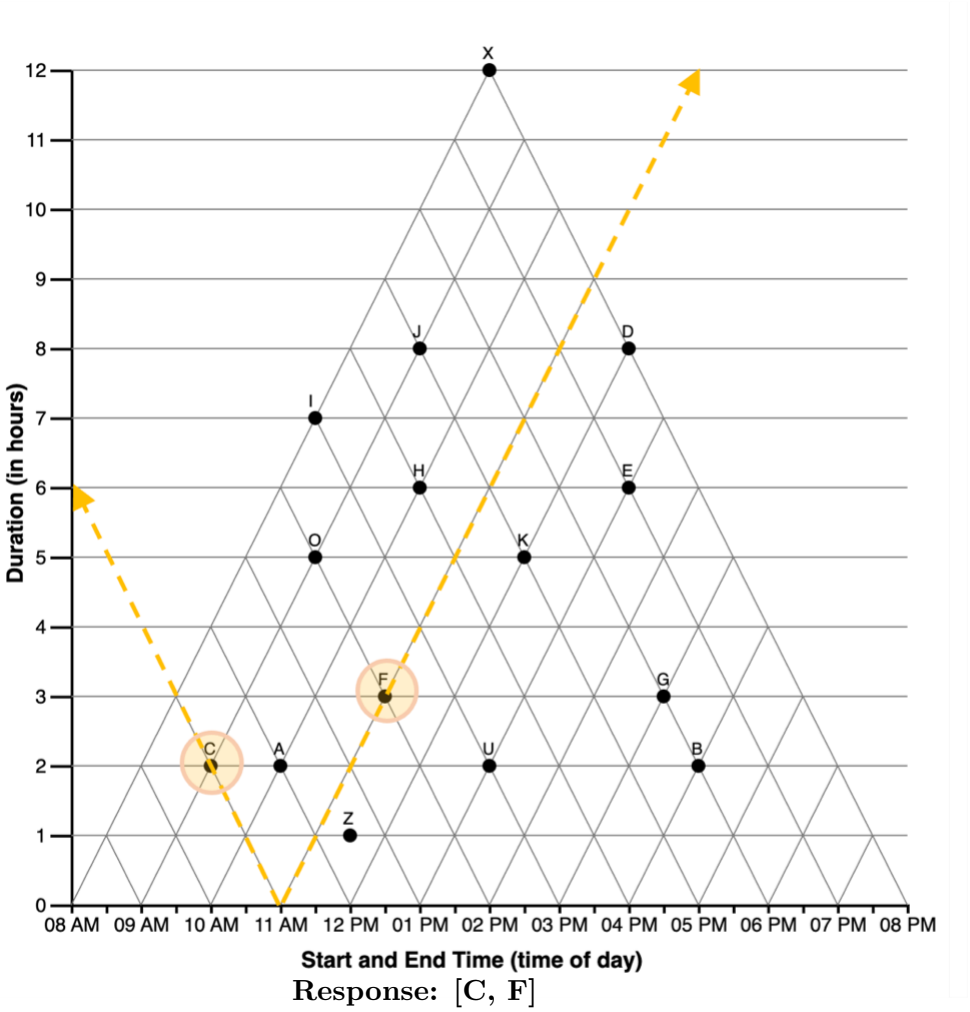
Which shifts start at 11am?



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**.

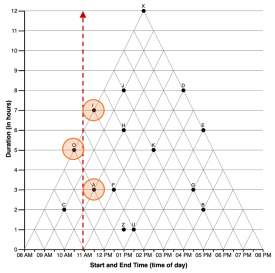
Which shifts start at
11am?



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?	

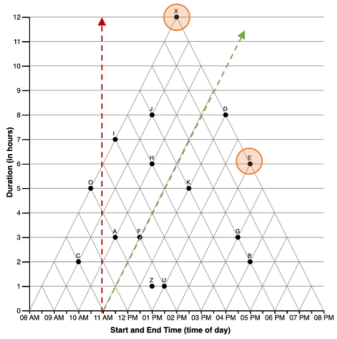


Responses: [AOI]

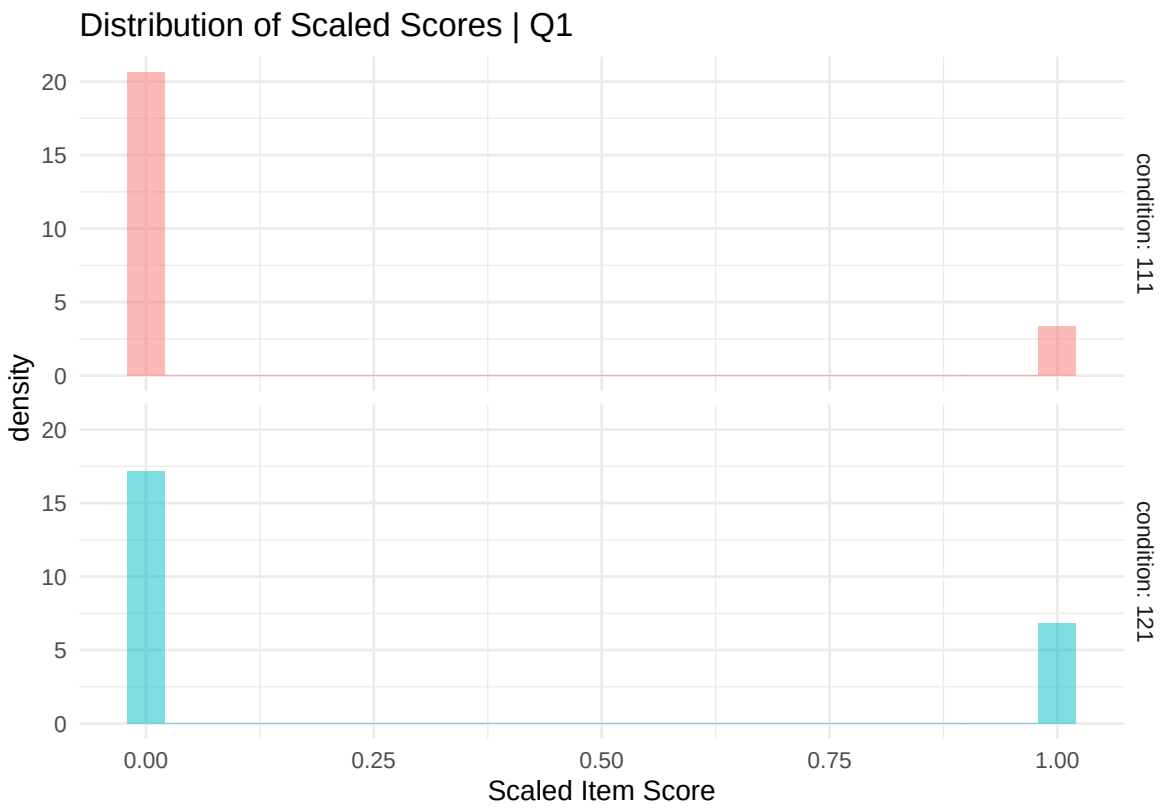
- indicates a **satisficing** strategy
- Consistent with the reader identifying the datapoints nearest to the orthogonal projection from the reference point point

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

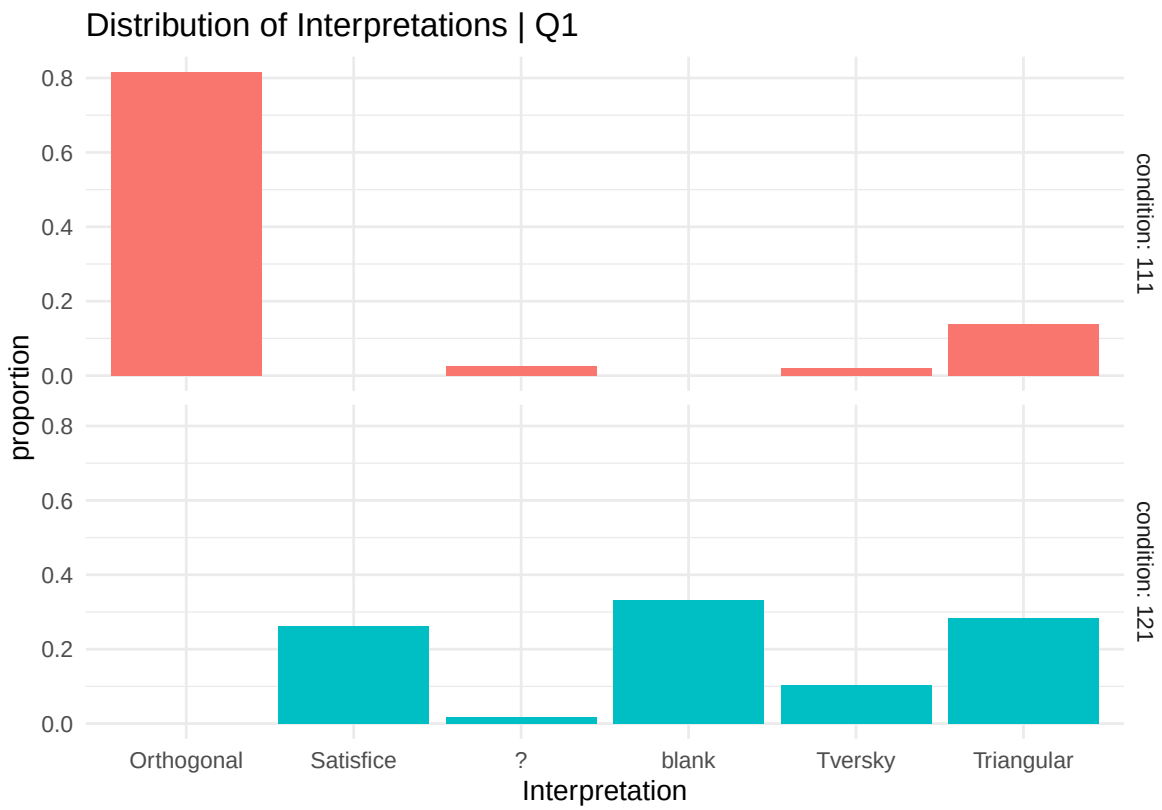
[E],[X]




```
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(~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")
```



2.4.1.2 Question #2

2.4.1.2.1 Q2. Control Condition

```
q <- keys_raw %>% filter(condition == "DEFAULT") %>% 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, "")
```

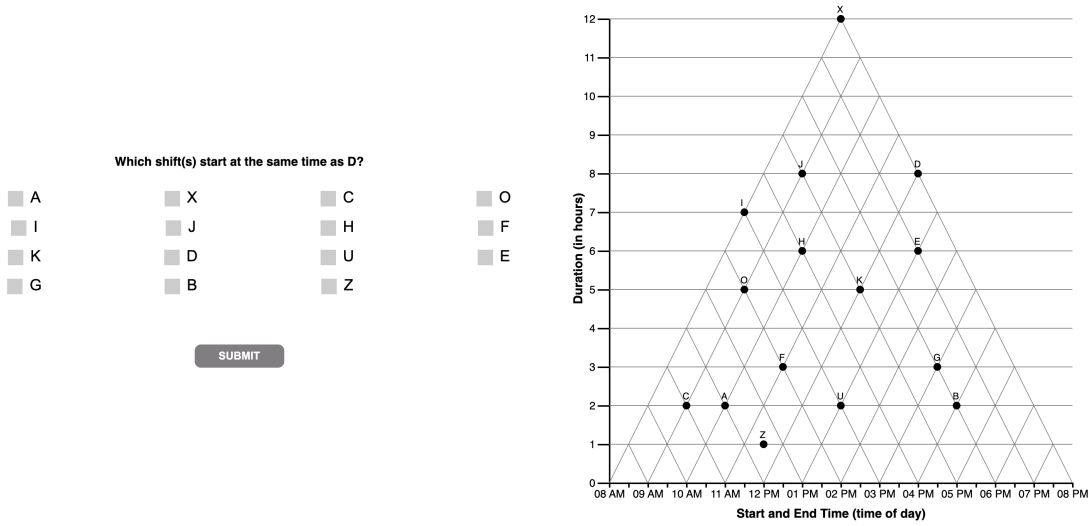


Figure 2.3: Q2—Control Condition

```
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(int2),
                    scaled = unique(score_SCALED)) %>%
  arrange(interpretation, desc(count)) %>%
```

Table 2.8: 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)}

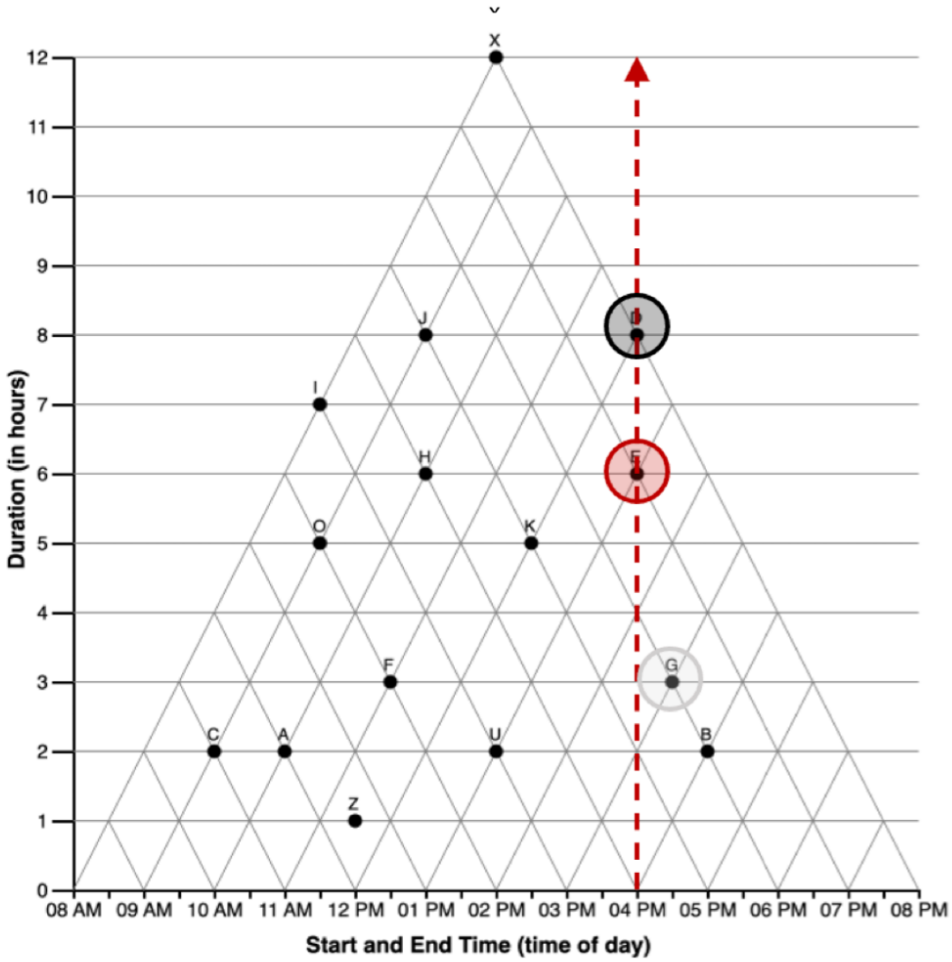
			Strict Score	Interpretation Scores				Discriminant
response	n	interpretation	absolute	tri	tversky	satisfice	orthogonal	scaled score
Triangular								
K	24	Triangular	1	1.000	0.500	NA	-0.083	1.0
DK	1	Triangular	1	1.000	0.500	NA	-0.083	1.0
Lines-Connect								
J	4	Tversky	0	-0.083	1.000	NA	-0.083	0.5
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	-0.5
C	1	?	0	-0.083	-0.077	NA	-0.083	-0.5

Note: n = number of responses in sample

\end{table}

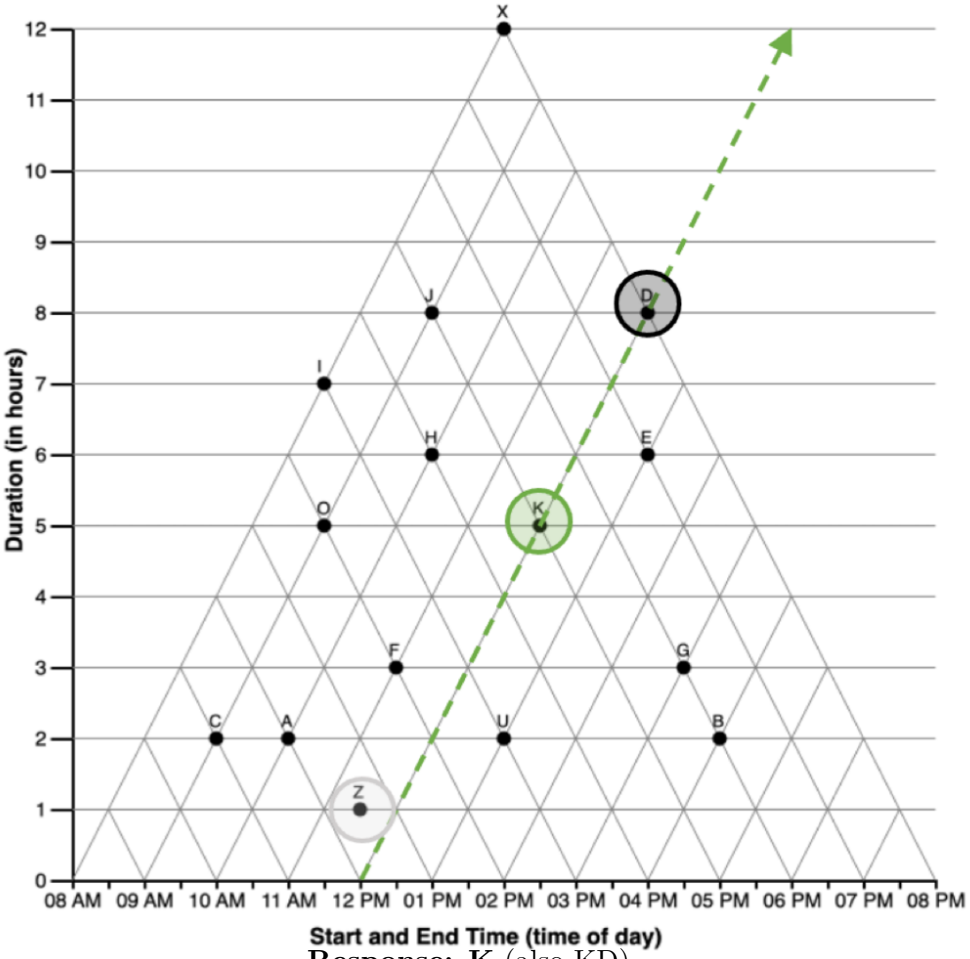
Again, we see that most subjects selected a response consistent with one of the identified interpretations. (note, when the question stem includes a data point rather than time as reference, we do not penalize respondents for selecting the reference data point *in addition* to an interpretation consistent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



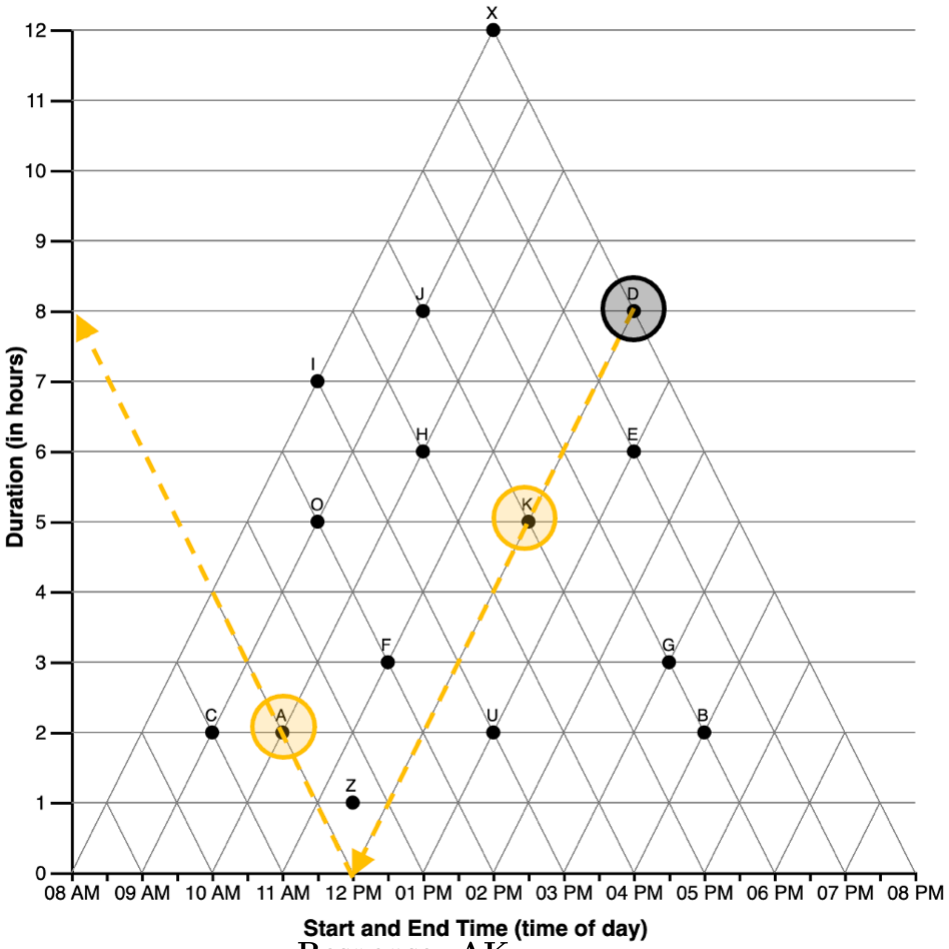
Reponse: **E** (also EG, DE)

- 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**.



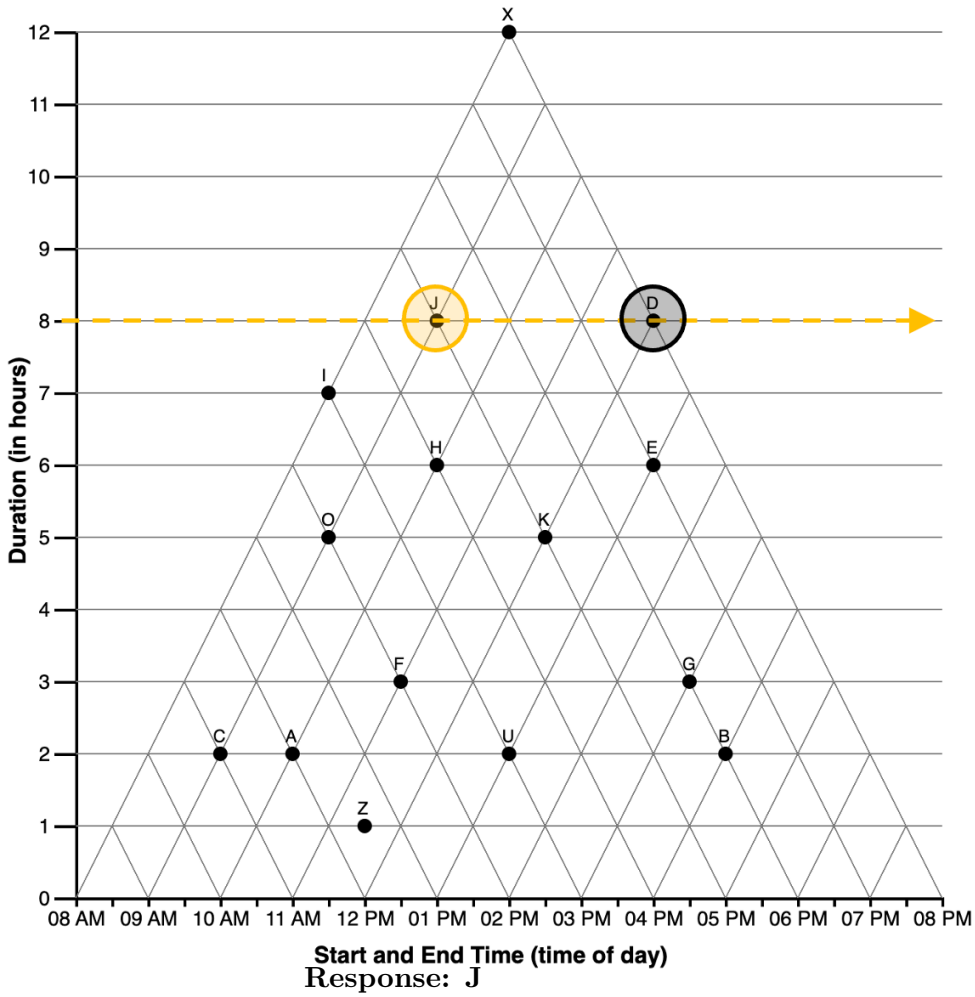
Response: **K** (also KD)

- 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**.



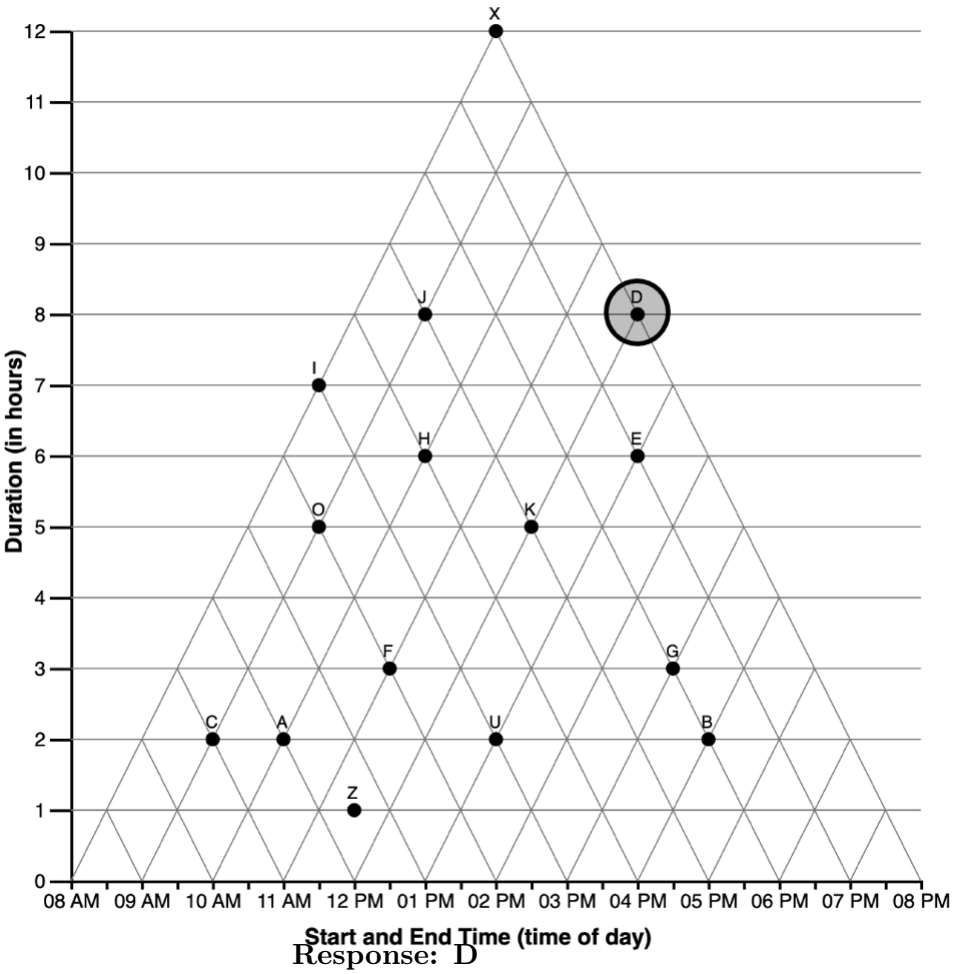
Response: AK

- 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

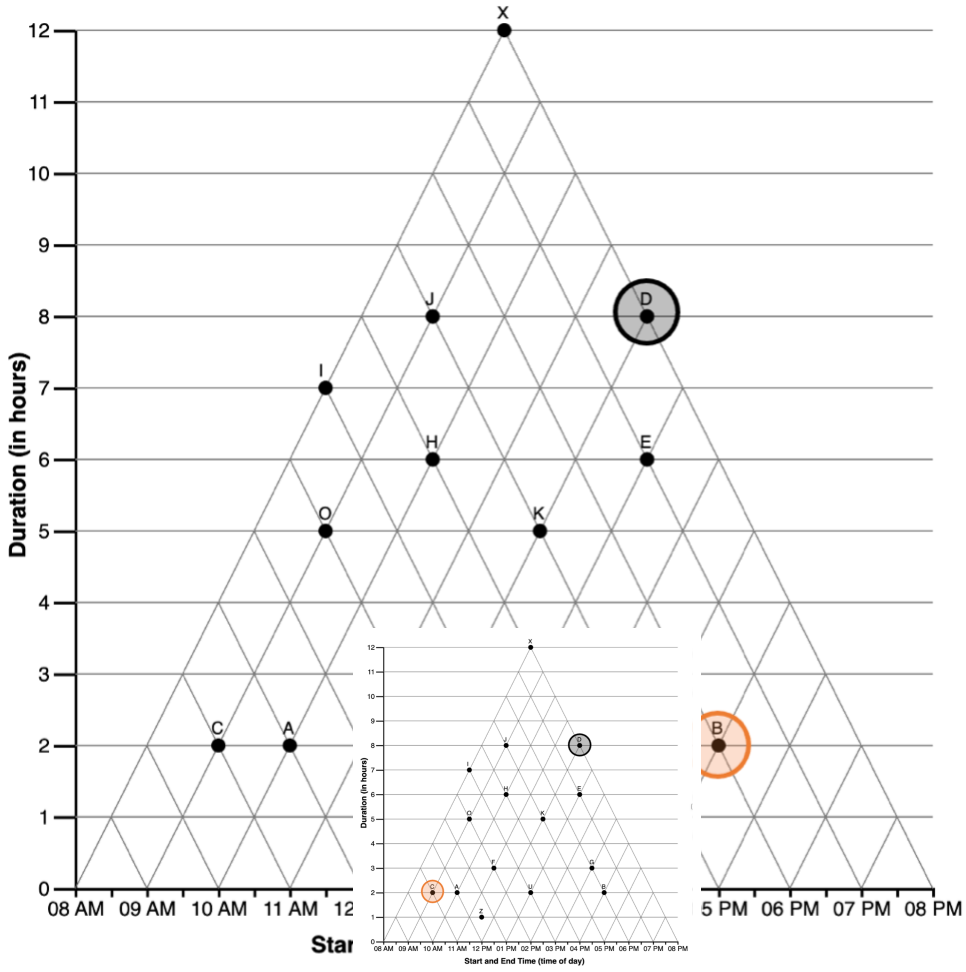
- indicates an **Tversky** strategy following connecting lines
- Consistent with the reader identifying the reference point (D) on the graph, and following its horizontal gridline to the y-axis, locating data point J.



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



2.4.1.2.2 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")
```

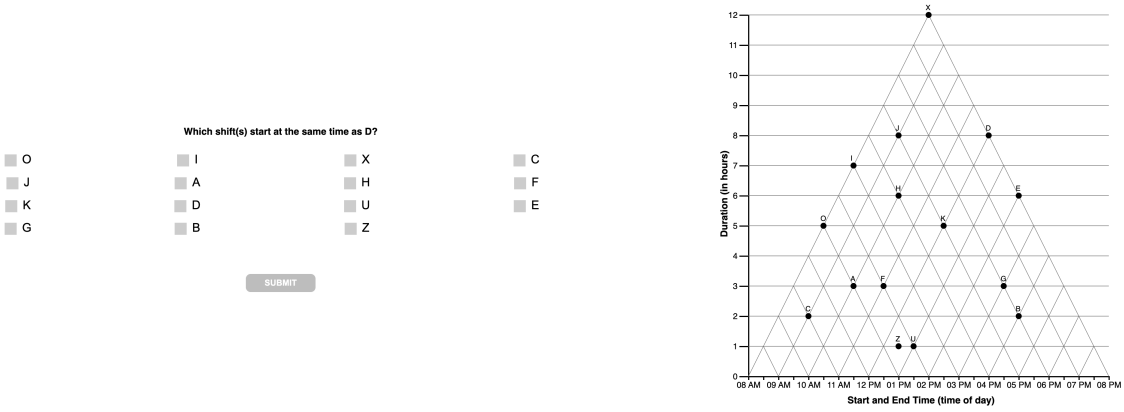


Figure 2.4: Q2—Impasse Condition

```
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(int2),
```

Table 2.11: 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}

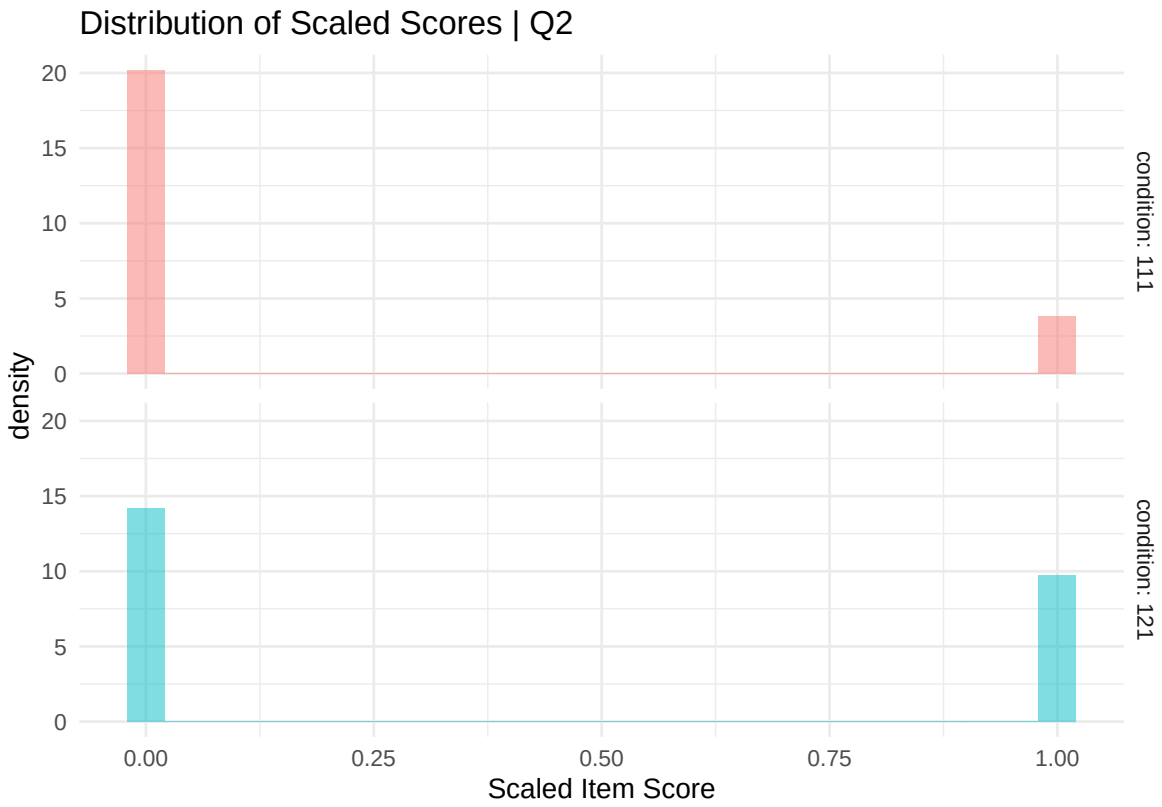
\caption{Frequency of Selected Response Options for Question #2 (Impasse Condition)}

response	n	interpretation	Strict Score	Interpretation Scores				Discrimi
			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	

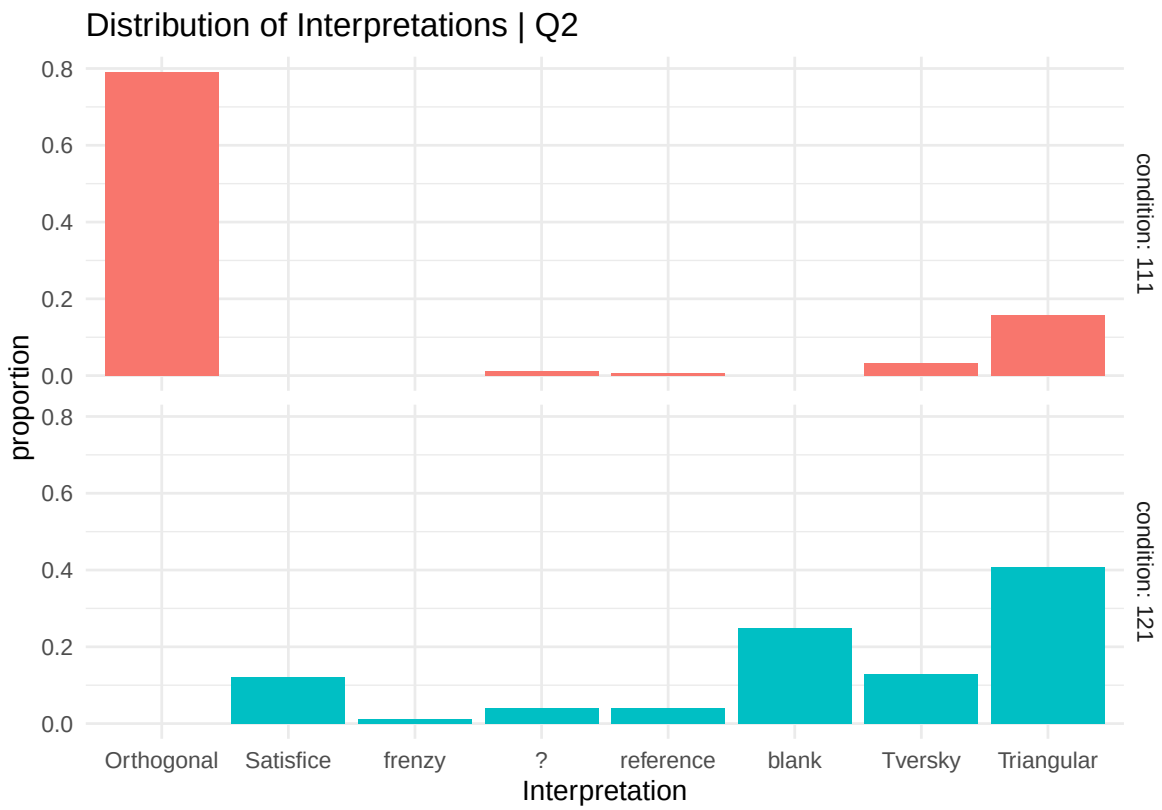
Note: n = number of responses in sample

\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(~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")
```



2.4.1.3 Question #3

2.4.1.3.1 Q3. Control Condition

```
q <- keys_raw %>% filter(condition == "DEFAULT") %>% 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, "")
```

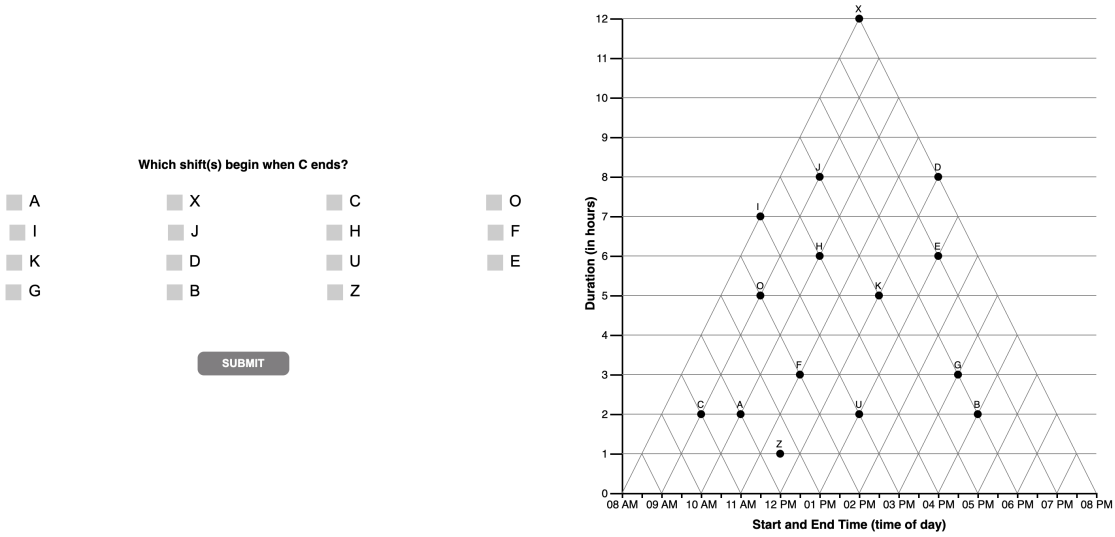


Figure 2.5: Q3—Control Condition

```
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(int2),
                    scaled = unique(score_SCALED)) %>%
```

Table 2.12: 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

```
  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)}