

SMRT

Chris Foster

2018-09-19

Contents

| | | |
|----------|---|-----------|
| 1 | Preface | 5 |
| 2 | Introduction to Smart Items | 7 |
| 2.1 | Purpose of Building Smart Items | 8 |
| 2.2 | Smart Item Logic | 8 |
| 3 | DOMC Difficulty Variance | 9 |
| 3.1 | Initial Run | 9 |
| 3.2 | Item 10B_v1 | 9 |
| 3.3 | Item 14b_v1 | 11 |
| 3.4 | Remove correct response | 13 |
| 3.5 | Item 10B_v1 | 13 |
| 3.6 | Item 14b_v1 | 14 |
| 3.7 | All Item Plots | 15 |
| 4 | DOMC Difficulty Factors | 17 |
| 5 | Total Test Score and Options Seen | 19 |
| 6 | Recall VS Deduction | 21 |
| 6.1 | Some Graphs | 22 |
| 7 | Final Words | 25 |

Chapter 1

Preface

This book is a **compilation** of different research pieces all compiled together into a single volume. The purpose of the volume is to provide a concise, research oriented view of the smart item and other accompanying item formats. Each chapter will be a different research topic and we will attempt to group similar research articles in close proximity to each other within the book.

Chapter 2

Introduction to Smart Items

A SmartItem is first and foremost an item, used on exams to measure important skills. Like traditional items, it has an ID number, is stored on a computer, is evaluated like traditional items with expert reviews, and eventually response data from examinees. It can be used on any kind of test design, such as CAT, LOFT, etc. If and when it doesn't function well, it can be repaired or deleted or retired.

The main difference between a smart item and a normal item is that the smart item is written with three areas of expertise: Subject matter, item writing and programming. With a well written and specific objective, with the help of a programmer and enough content expertise it is possible to write a single item which can cover the entire range of an objective. With the help of a programmer an item writer can write a series of stems, options, correct responses and incorrect responses that can generate a large amount of potential item derivatives based on a single objective. This process creates an item that is less static than a single multiple choice item.

In order better understand a smart item it is best to start with an example. An illustrative example would come from an elementary math test. A single objective might be: Add two single digit numbers. There are only 10 single digit numbers (including 0) So really there is only $(10! / 2!(10-2)!) = 45$ possible options as long as order doesn't matter.

Now, a single item writer could write all 45 items and cover the objective completely. However, it is also possible to write a simple program which generates all 45 possible questions. Now, for a fixed form test it would be likely that the item writer would not write all 45 options but instead write 2 or 3 of which one would be selected for the first form of the test while a different one might be selected for a second form. However, when administering a smart item to participants each participant would get a random stem and random options (including the correct option).

Now, for a simple math objective it might not be necessary to write an algorithm that writes the 45 different possible stems for the objective. However, imagine an objective where there are 206 possible answers such as "Identify each bone in the human body." Or perhaps there is an objective which asks participants to arrange 4 words in alphabetical order. The words can be anything in the human dictionary. Now there are 170,000 words in the english language and picking 4 leaves 3.479×10^{19} possible options to completely cover the objective content and no item writer can write all of them and given current test construction methods there is no reason to do so.

2.1 Purpose of Building Smart Items

2.2 Smart Item Logic

A first step to understanding the logic behind smart items is to understand the logic of randomization in experimentation. Sir Ronald Fisher (1925) outlined what is considered the cornerstone of experimental research today: randomization. Randomization has three primary purposes:

- 1) It helps to distribute idiosyncratic characteristics of participants to groups so that it does not bias the outcome. If participants could self select groups or were grouped based on characteristics than it could create systematic biases in the outcome based on participant characteristics.
- 2) Randomization helps calculate unbiased estimate of error effects. IE: those effects not attributable to the manipulation of an independent variable
- 3) Randomization helps ensure that error effects are statistically independent.

Now, considering point #1 a bit more: Randomization helps ensure that within group variability is Independent and identically distributed (IID) or in other words, within group variability does not contain bias and is simply noise. Without randomization it could easily contain any number of biases which could decrease or increase the differences between groups. It is impossible to list all possible systematic biases that could creep into an experiment. Maybe all college educated participants self select themselves into a specific group or one gender reacts differently to a group assignment than another.

While other papers have talked in length about the importance of randomization in experimental design for the purposes of this section randomization removes systematic bias within group.

One natural artifact of the randomization process is an increase in within-group variation. If participants are assigned to groups based on characteristics or allowed to self select, more similar participants will end up in the same group reducing the amount of variability in the group. While a decrease in within-group variability inevitably increases the probability of a significant effect in an experiment, the significant effect may simply be due to a bias brought by the selection process... which simply shows the importance of randomization. Even though variation is introduced, results are more trustworthy.

Chapter 3

DOMC Difficulty Variance

3.1 Initial Run

Here is a document showing the results of item families derived from a single DOMC stem. Essentially we treat each possible combination of options as a different question just to see the amount of variance from a single DOMC stem.

In this first run we treat each different option combination as a different item, including those for people who never saw the correct response (making all these p-values 0)

```
library(dplyr)
library(readr)
library(knitr)
library(ggplot2)
library(lemon)

setwd_thisdir <- function () {
  this.dir <- dirname(parent.frame(3)$ofile)
  setwd(this.dir)
}

hp_data = read_csv('data/domc_order_difficulty/Full Responses After Exclusions.csv')

hp_clean = hp_data %>% filter(item_type == 'domc', item_component_type == 'domc_option') %>% group_by(d
hp_summary = hp_clean %>% group_by(delivery_id, item_id) %>% summarize(item_total_seconds = max(item_to
hp_items = hp_summary %>% group_by(item_id, order) %>% summarize(p_value = mean(score), count = n())
```

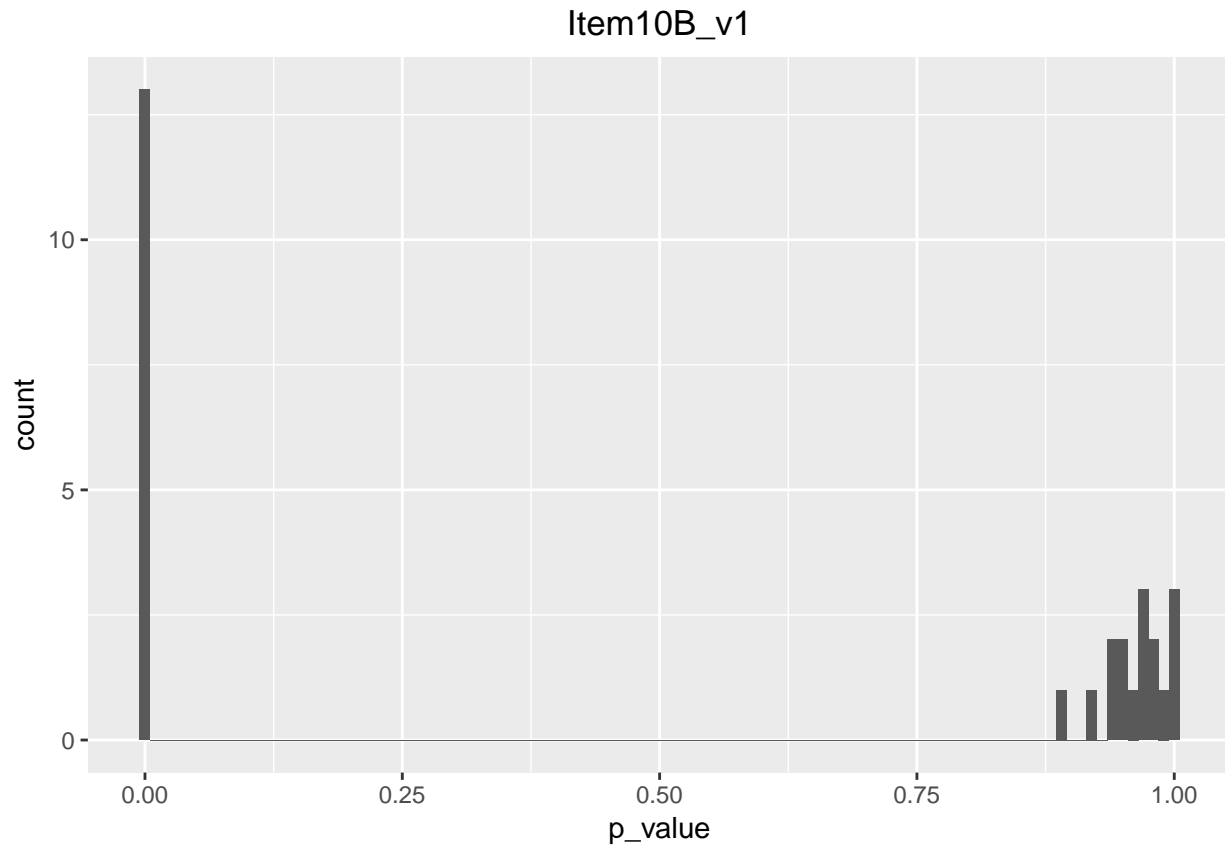
3.2 Item 10B__v1

```
tenb_v1 = hp_items %>% filter(item_id == '10B_v1')
kable(tenb_v1)
```

| item_id | order | p_value | count |
|---------|-------|-----------|-------|
| 10B_v1 | 0 | 0.8950000 | 200 |
| 10B_v1 | 01 | 1.0000000 | 42 |
| 10B_v1 | 012 | 0.9677419 | 31 |
| 10B_v1 | 0123 | 1.0000000 | 40 |
| 10B_v1 | 01234 | 0.9876543 | 162 |
| 10B_v1 | 0124 | 0.9827586 | 58 |
| 10B_v1 | 013 | 0.9666667 | 30 |
| 10B_v1 | 0134 | 1.0000000 | 45 |
| 10B_v1 | 014 | 0.9629630 | 27 |
| 10B_v1 | 02 | 0.9365079 | 63 |
| 10B_v1 | 023 | 0.9487179 | 39 |
| 10B_v1 | 0234 | 0.9354839 | 62 |
| 10B_v1 | 024 | 0.9210526 | 38 |
| 10B_v1 | 03 | 0.9464286 | 56 |
| 10B_v1 | 034 | 0.9666667 | 30 |
| 10B_v1 | 04 | 0.9756098 | 41 |
| 10B_v1 | 1 | 0.0000000 | 21 |
| 10B_v1 | 12 | 0.0000000 | 6 |
| 10B_v1 | 123 | 0.0000000 | 8 |
| 10B_v1 | 1234 | 0.0000000 | 13 |
| 10B_v1 | 124 | 0.0000000 | 3 |
| 10B_v1 | 13 | 0.0000000 | 10 |
| 10B_v1 | 134 | 0.0000000 | 7 |
| 10B_v1 | 14 | 0.0000000 | 8 |
| 10B_v1 | 2 | 0.0000000 | 3 |
| 10B_v1 | 234 | 0.0000000 | 1 |
| 10B_v1 | 24 | 0.0000000 | 4 |
| 10B_v1 | 3 | 0.0000000 | 1 |
| 10B_v1 | 4 | 0.0000000 | 2 |

3.2.1 Histogram

```
theme_update(plot.title = element_text(hjust = 0.5))
ggplot(tenb_v1, aes(x=p_value)) + geom_histogram(binwidth=.01) + labs(title = "Item10B_v1")
```



3.3 Item 14b_v1

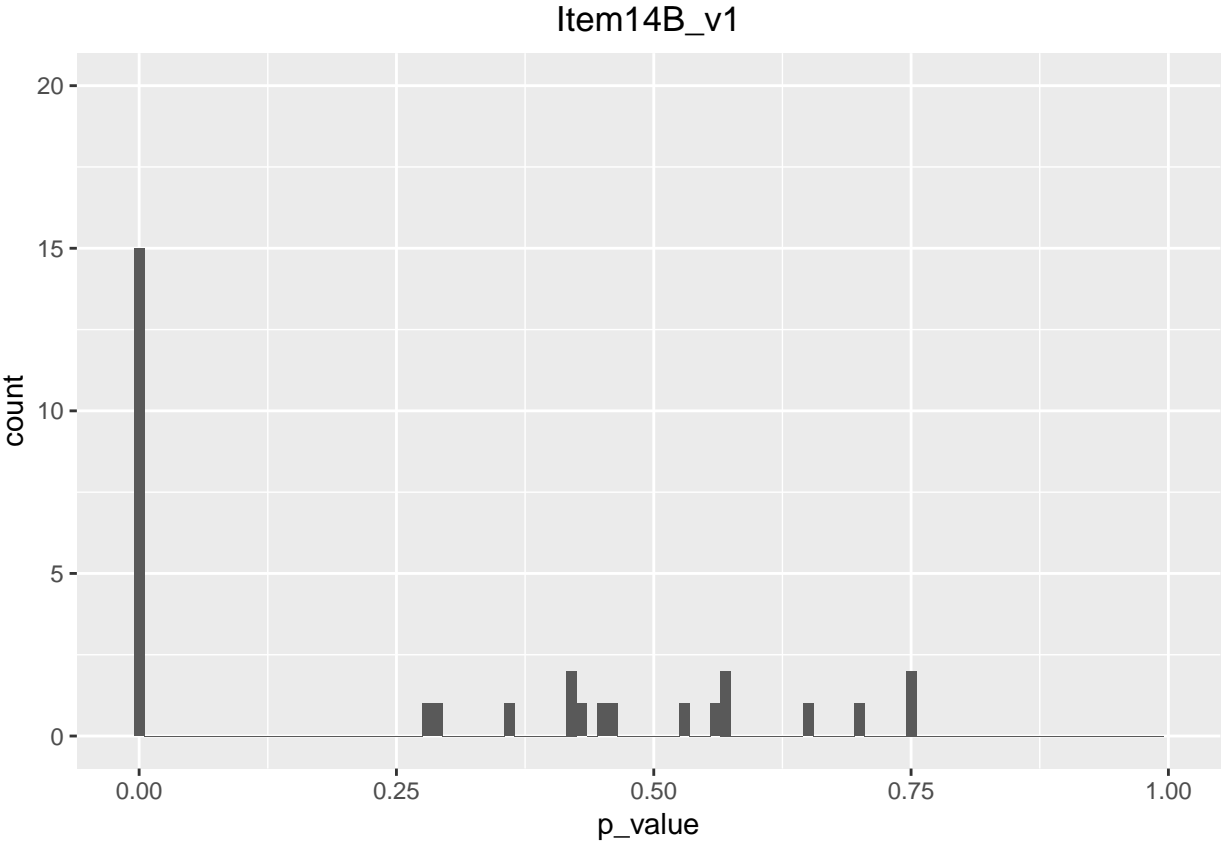
```
fourteenb_v1 = hp_items %>% filter(item_id == '14B_v1')
kable(fourteenb_v1, format = "markdown")
```

| item_id | order | p_value | count |
|---------|-------|-----------|-------|
| 14B_v1 | 0 | 0.4235294 | 170 |
| 14B_v1 | 01 | 0.2765957 | 47 |
| 14B_v1 | 012 | 0.3636364 | 11 |
| 14B_v1 | 0123 | 0.2941176 | 17 |
| 14B_v1 | 01234 | 0.7049180 | 61 |
| 14B_v1 | 0124 | 0.7500000 | 12 |
| 14B_v1 | 013 | 0.5652174 | 23 |
| 14B_v1 | 0134 | 0.5666667 | 30 |
| 14B_v1 | 014 | 0.4583333 | 24 |
| 14B_v1 | 02 | 0.5263158 | 19 |
| 14B_v1 | 023 | 0.7500000 | 16 |
| 14B_v1 | 0234 | 0.4545455 | 11 |
| 14B_v1 | 024 | 0.6470588 | 17 |
| 14B_v1 | 03 | 0.4181818 | 55 |
| 14B_v1 | 034 | 0.5555556 | 36 |
| 14B_v1 | 04 | 0.4333333 | 30 |

| item_id | order | p_value | count |
|---------|-------|-----------|-------|
| 14B_v1 | 1 | 0.0000000 | 39 |
| 14B_v1 | 12 | 0.0000000 | 23 |
| 14B_v1 | 123 | 0.0000000 | 22 |
| 14B_v1 | 1234 | 0.0000000 | 36 |
| 14B_v1 | 124 | 0.0000000 | 19 |
| 14B_v1 | 13 | 0.0000000 | 8 |
| 14B_v1 | 134 | 0.0000000 | 11 |
| 14B_v1 | 14 | 0.0000000 | 9 |
| 14B_v1 | 2 | 0.0000000 | 129 |
| 14B_v1 | 23 | 0.0000000 | 35 |
| 14B_v1 | 234 | 0.0000000 | 21 |
| 14B_v1 | 24 | 0.0000000 | 20 |
| 14B_v1 | 3 | 0.0000000 | 30 |
| 14B_v1 | 34 | 0.0000000 | 11 |
| 14B_v1 | 4 | 0.0000000 | 59 |

3.3.1 Histogram

```
ggplot(fourteenb_v1, aes(x=p_value)) + geom_histogram(binwidth=.01) + labs(title = "Item14B_v1")+ xlim(
```



3.4 Remove correct response

We notice quickly that it is pretty difficult to interpret the p-values given that there is a significant amount of people who never see the correct answer so the p-value for those “children” is 0. Here is what happens when we redefine what it means to be a different item. Instead of looking at every combination of seen options we are only going to look at distractors. This means that someone who saw the correct option and the first option (coded 01) will have “seen” the “same” question as someone who just saw the first distractor(coded 1).

It cleans things up a bit but it is important to recognize what is going on.

```
hp_summary$order_no_cr = gsub('0','',hp_summary$order)
hp_summary$order_no_cr = ifelse(hp_summary$order_no_cr == '', '0', hp_summary$order_no_cr)
hp_items_no_cr = hp_summary %>% group_by(item_id, order_no_cr) %>% summarize(p_value = mean(score), count = count())
```

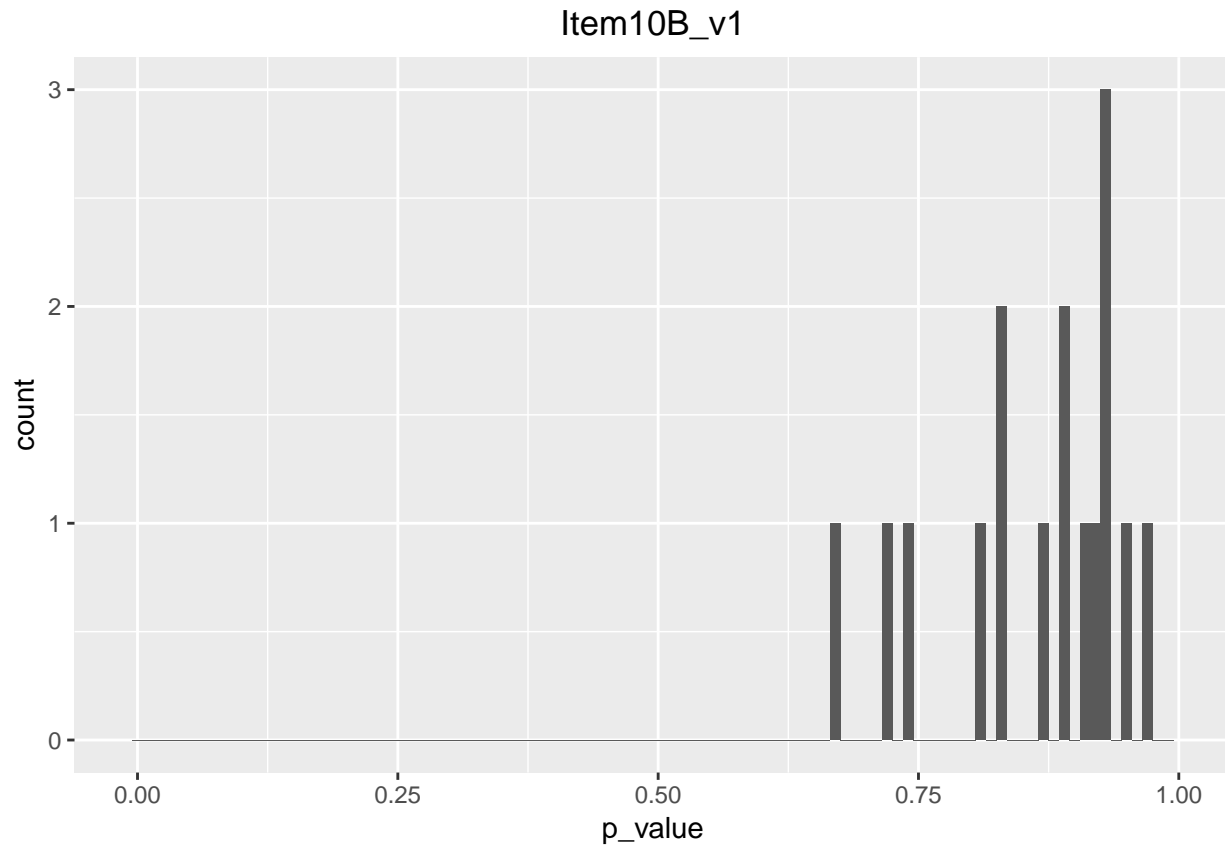
3.5 Item 10B_v1

```
tenb_v1 = hp_items_no_cr %>% filter(item_id == '10B_v1')
kable(tenb_v1)
```

| item_id | order_no_cr | p_value | count |
|---------|-------------|-----------|-------|
| 10B_v1 | 0 | 0.8950000 | 200 |
| 10B_v1 | 1 | 0.6666667 | 63 |
| 10B_v1 | 12 | 0.8108108 | 37 |
| 10B_v1 | 123 | 0.8333333 | 48 |
| 10B_v1 | 1234 | 0.9142857 | 175 |
| 10B_v1 | 124 | 0.9344262 | 61 |
| 10B_v1 | 13 | 0.7250000 | 40 |
| 10B_v1 | 134 | 0.8653846 | 52 |
| 10B_v1 | 14 | 0.7428571 | 35 |
| 10B_v1 | 2 | 0.8939394 | 66 |
| 10B_v1 | 23 | 0.9487179 | 39 |
| 10B_v1 | 234 | 0.9206349 | 63 |
| 10B_v1 | 24 | 0.8333333 | 42 |
| 10B_v1 | 3 | 0.9298246 | 57 |
| 10B_v1 | 34 | 0.9666667 | 30 |
| 10B_v1 | 4 | 0.9302326 | 43 |

3.5.1 Histogram

```
theme_update(plot.title = element_text(hjust = 0.5))
ggplot(tenb_v1, aes(x=p_value)) + geom_histogram(binwidth=.01) + labs(title = "Item10B_v1") + xlim(-.01, 1)
```



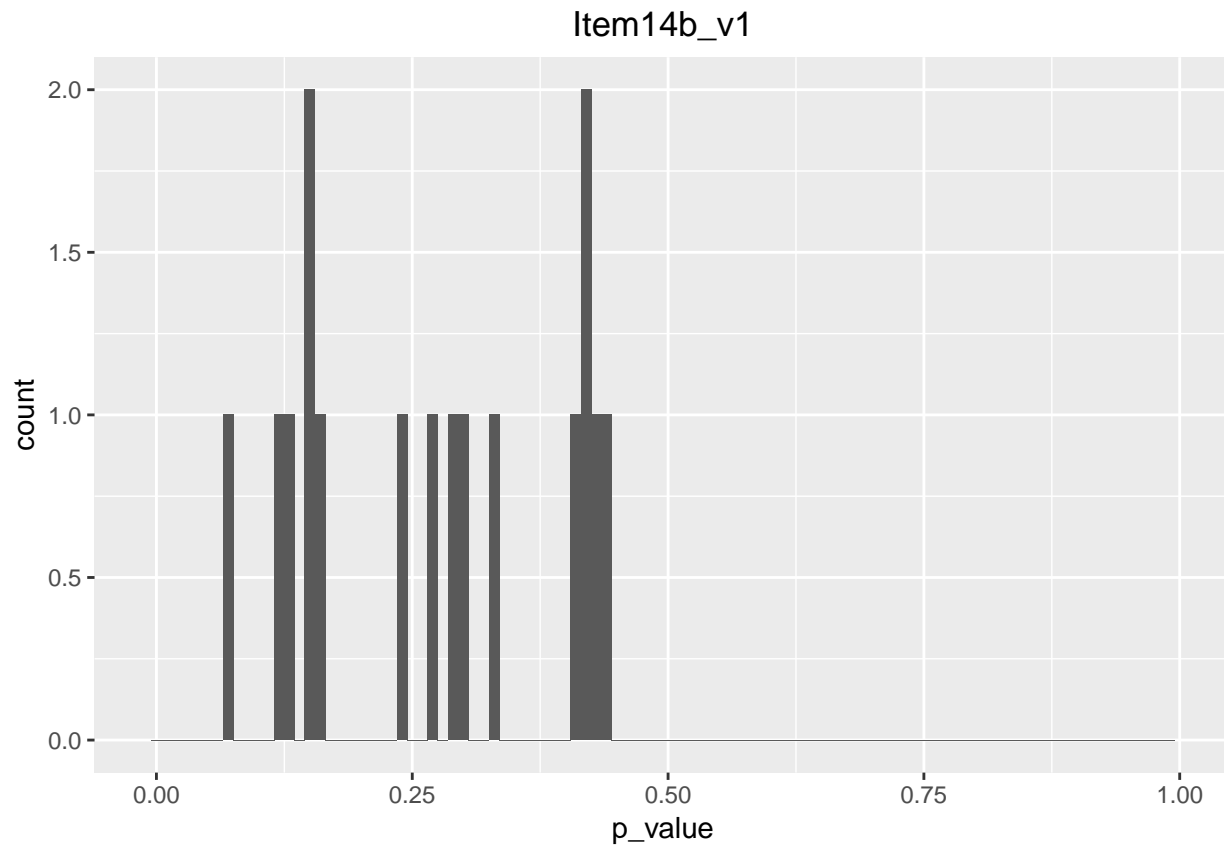
3.6 Item 14b_v1

```
fourteenb_v1 = hp_items_no_cr %>% filter(item_id == '14B_v1')
kable(fourteenb_v1)
```

| item_id | order_no_cr | p_value | count |
|---------|-------------|-----------|-------|
| 14B_v1 | 0 | 0.4235294 | 170 |
| 14B_v1 | 1 | 0.1511628 | 86 |
| 14B_v1 | 12 | 0.1176471 | 34 |
| 14B_v1 | 123 | 0.1282051 | 39 |
| 14B_v1 | 1234 | 0.4432990 | 97 |
| 14B_v1 | 124 | 0.2903226 | 31 |
| 14B_v1 | 13 | 0.4193548 | 31 |
| 14B_v1 | 134 | 0.4146341 | 41 |
| 14B_v1 | 14 | 0.3333333 | 33 |
| 14B_v1 | 2 | 0.0675676 | 148 |
| 14B_v1 | 23 | 0.2352941 | 51 |
| 14B_v1 | 234 | 0.1562500 | 32 |
| 14B_v1 | 24 | 0.2972973 | 37 |
| 14B_v1 | 3 | 0.2705882 | 85 |
| 14B_v1 | 34 | 0.4255319 | 47 |
| 14B_v1 | 4 | 0.1460674 | 89 |

3.6.1 Histogram

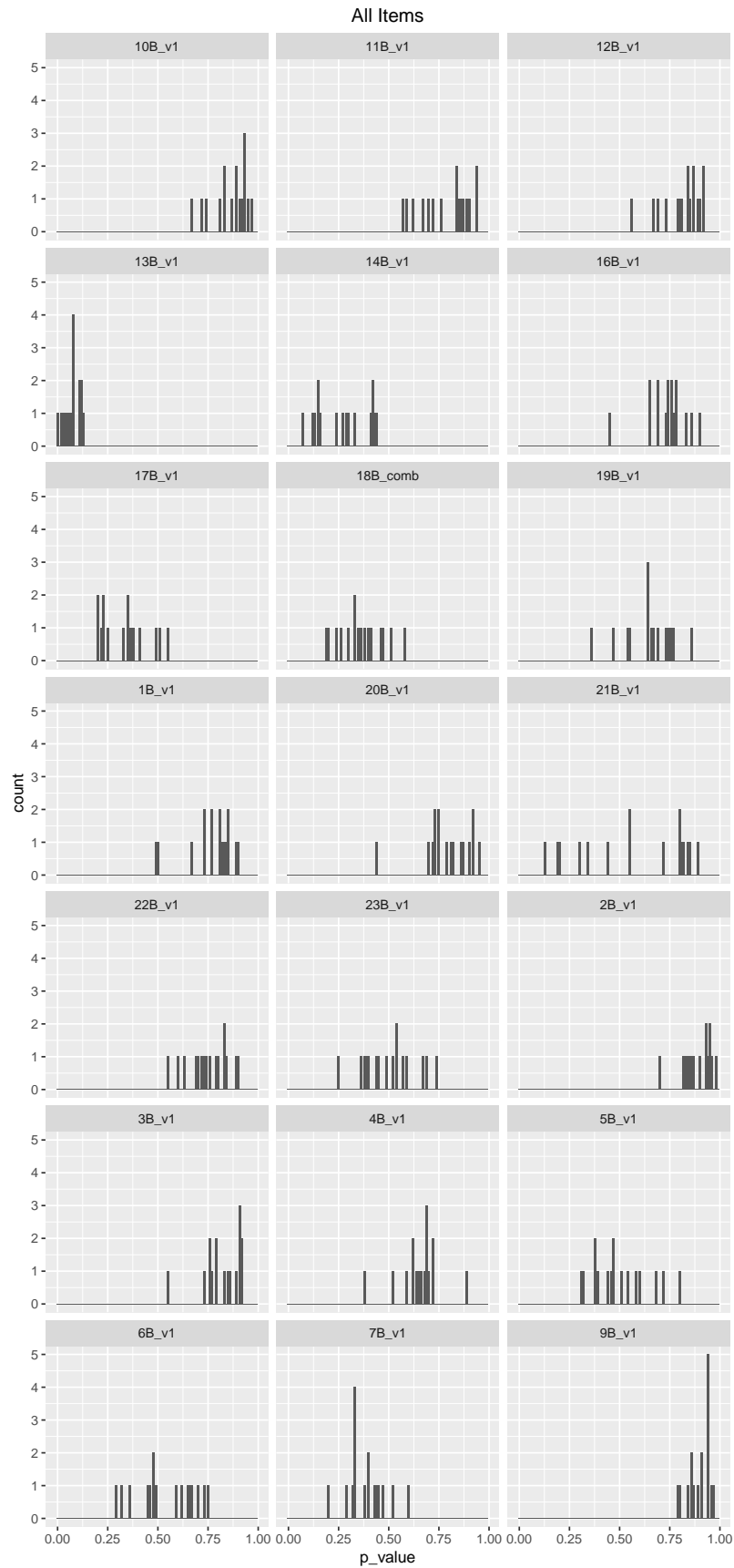
```
ggplot(fourteenb_v1, aes(x=p_value)) + geom_histogram(binwidth=.01) + labs(title = "Item14b_v1") + xlim
```



3.7 All Item Plots

```
knitr::opts_chunk$set(fig.width=12, fig.height=20)

ggplot(hp_items_no_cr, aes(x=p_value)) + geom_histogram(binwidth=.01) + labs(title = "All Items") + xlim
```



Chapter 4

DOMC Difficulty Factors

The purpose of this chapter is to briefly address portions of the item format that may lead to systematic differences in difficulty for DOMC items

```
library(dplyr)
library(readr)
library(knitr)
library(ggplot2)
library(lemon)

setwd_thisdir <- function () {
  this.dir <- dirname(parent.frame(3)$ofile)
  setwd(this.dir)
}

hp_data = read_csv('data/domc_order_difficulty/Full Responses After Exclusions.csv')
```

```
## Parsed with column specification:
## cols(
##   delivery_id = col_character(),
##   item_id = col_character(),
##   item_type = col_character(),
##   item_total_seconds = col_double(),
##   score = col_integer(),
##   points_possible = col_integer(),
##   key = col_character(),
##   option_presented = col_character(),
##   response = col_integer(),
##   item_component_type = col_character(),
##   item_component_seconds = col_double(),
##   start_time = col_datetime(format = ""),
##   end_time = col_datetime(format = "")
## )
```

```
hp_clean = hp_data %>% filter(item_type == 'domc' | item_type == 'multiple_choice', item_component_type == 'multiple_choice')
hp_summary = hp_clean %>% group_by(delivery_id, item_id, item_type) %>% summarize(item_total_seconds = sum(item_total_seconds))
```

```
total_score = hp_summary %>% group_by(delivery_id) %>% summarize(total_score = sum(score))  
hp_final = left_join(hp_summary, total_score)  
  
## Joining, by = "delivery_id"  
  
hp_final$char_count = nchar(hp_final$order)
```

Chapter 5

Total Test Score and Options Seen

Simply running a logistic regression predicting if a person will get an item correct based on the total number of options seen seems disingenuous for a DOMC item. The reason being is that more proficient examinees will on average see more options as they are more likely to reject incorrect options.

Here we look at some regression results

```
mylogit <- glm(score ~ total_score + factor(item_type) + char_count, data = hp_final, family = "binomial")
summary(mylogit)
```

```
##
## Call:
## glm(formula = score ~ total_score + factor(item_type) + char_count,
##      family = "binomial", data = hp_final)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.4972  -1.0727   0.6177   0.8451   2.3305
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.587105   0.055005  -65.21  <2e-16 ***
## total_score      0.217796   0.003298   66.04  <2e-16 ***
## factor(item_type)multiple_choice -6.181737   0.160348  -38.55  <2e-16 ***
## char_count       0.504366   0.012474   40.43  <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: 59053  on 45758  degrees of freedom
## Residual deviance: 51520  on 45755  degrees of freedom
## AIC: 51528
##
## Number of Fisher Scoring iterations: 4
```

```
mylogit <- glm(score ~ factor(item_type) + char_count + factor(item_type)*char_count, data = hp_final, family = "binomial")
summary(mylogit)
```

```
##
## Call:
## glm(formula = score ~ factor(item_type) + char_count + factor(item_type) *
##      char_count, family = "binomial", data = hp_final)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.0489  -1.3260   0.8314   0.8926   1.2630
##
## Coefficients: (1 not defined because of singularities)
##
##              Estimate Std. Error z value
## (Intercept)      -0.74081    0.03029  -24.46
## factor(item_type)multiple_choice -6.67167    0.15294  -43.62
## char_count         0.54180    0.01188   45.61
## factor(item_type)multiple_choice:char_count      NA         NA      NA
##
##              Pr(>|z|)
## (Intercept)      <2e-16 ***
## factor(item_type)multiple_choice      <2e-16 ***
## char_count        <2e-16 ***
## factor(item_type)multiple_choice:char_count      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 59053  on 45758  degrees of freedom
## Residual deviance: 56570  on 45756  degrees of freedom
## AIC: 56576
##
## Number of Fisher Scoring iterations: 4
```

Chapter 6

Recall VS Deduction

In this chapter we will look at some differences between recall and deduction items on a Harry Potter assessment.

```
library(dplyr)
library(readr)
library(knitr)
library(ggplot2)
library(lemon)
library(stringr)
setwd_thisdir <- function () {
  this.dir <- dirname(parent.frame(3)$ofile)
  setwd(this.dir)
}

hp_data = read_csv('data/domc_order_difficulty/Full Responses After Exclusions.csv')
```

```
## Parsed with column specification:
## cols(
##   delivery_id = col_character(),
##   item_id = col_character(),
##   item_type = col_character(),
##   item_total_seconds = col_double(),
##   score = col_integer(),
##   points_possible = col_integer(),
##   key = col_character(),
##   option_presented = col_character(),
##   response = col_integer(),
##   item_component_type = col_character(),
##   item_component_seconds = col_double(),
##   start_time = col_datetime(format = ""),
##   end_time = col_datetime(format = "")
## )
```

```
hp_clean = hp_data %>% filter(item_type == 'domc' | item_type == 'multiple_choice', item_component_type == 'recall')

recall <- c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12")
deduction = c("13", "14", "15", "16", "17", "18", "19", "20", "21", "22")
```

```

hp_clean$deduction <- ifelse(grepl(paste(deduction,collapse="|"), hp_clean$item_id),1,0)

hp_items = hp_clean %>% group_by(item_id, item_type) %>% summarize(p_value = mean(score), count = n(), )
hp_items$item_number = str_extract(hp_items$item_id, "[0-9]+")

```

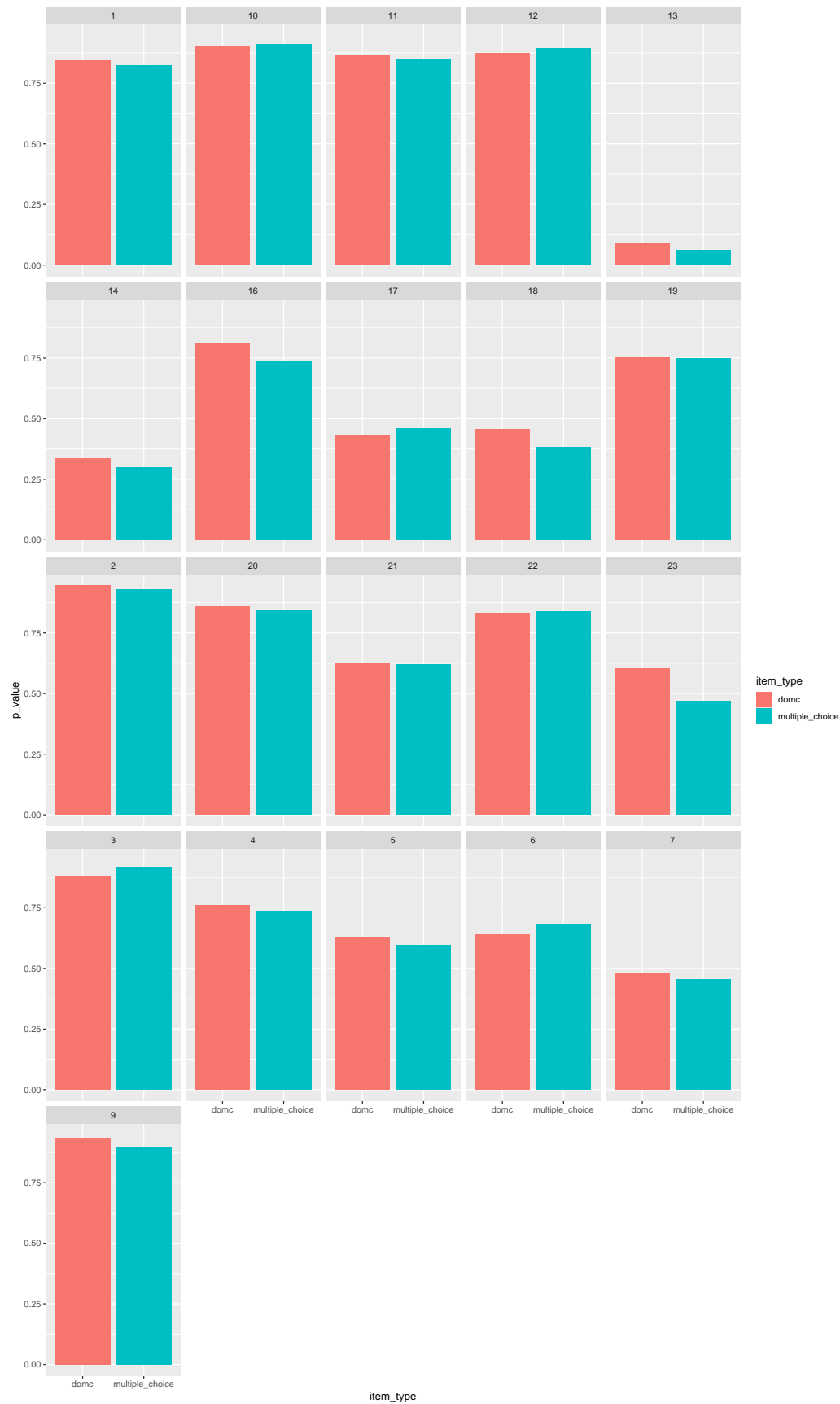
6.1 Some Graphs

Here is the difference in p-value between DOMC and multiple choice for the same items.

```

ggplot(hp_items, aes(x=item_type, y=p_value, fill=item_type)) + geom_bar(stat="identity") + facet_wrap

```



Here is the difference in p-value between DOMC and multiple choice for the same items.

```
just_mc = hp_items %>% filter(item_type == 'multiple_choice')
just_domc = hp_items %>% filter(item_type != 'multiple_choice')

differences = hp_items %>% group_by(item_type) %>% summarize(average_p_value = mean(p_value, na.rm=TRUE))
kable(differences)
```

| item_type | average_p_value |
|-----------------|-----------------|
| domc | 0.6933272 |
| multiple_choice | 0.6741205 |

The total difference in p-values between item_types is -0.0192067.

Multiple Choice items appear to be slightly more difficult.

Chapter 7

Final Words

We have finished a nice book.

Bibliography

Fisher, R. (1925). *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh, Tweeddale Court, 1st edition.