Q-Matrix

Discovering knowledge & skills required in a test

Eddie Lin

2018/03/17

Project Description

Q-matrix in education is often used in test design and development. As far as I know, it is an important technique in psychometrics and useful to examine the validity of a test. To put it simply(in my own terms), I will say it is used to “see how many (and what) knowledge concepts our test is testing students, and how if we should revise the test design based on its error score ”.

There is a great introduction of Q-matrix by Dr. Ryan Baker, which is useful for people who want to know more about the development and application of this technique

This project's objectives are:

1. Create a Q-matrix to with self-defined knowledge concepts to measure the validity of a test design

2. Review the error score of the Q-matrix & suggest ways to revise the test

Tools & Data

- R
- R packages: e1071
- Data source: test result of 29 students after a take-home exam using R language

Data Description

This data set contains information of students' responses to 14 test questions. There are 3 types of responses:

- Yes: the student got the answer correctly
- No: the student got the answer wrongly
- Did not answer

This is "NOT" a self-report data. All the responses were generated by computer system after it matched students' answers to each question to the correct answers

Build an initial Q-matrix

- We will start by building a Q-Matrix from the questions on the 14-question test
- Based on the questions, I've defined 4 concepts which each question may/may not require

```
DF1 <- read.csv("Q-matrix-formative1-eddie.csv", header = TRUE)
DF1[ , 1] <- NULL
rownames(DF1) <- c("q1","q2","q3","q4","q5","q6","q7","q8","q9","q10'
colnames(DF1) <- c("c1","c2","c3","c4")
DF1 <- as.matrix(DF1)
```

- The 4 concepts I defined are:
  - c1: R basic commands
  - c2: data manipulation
  - c3: data reshaping
  - c4: matrix concept

Define concept states for your Q-Matrix

- Define all possible concept states for your Q-Matrix
- A concept state is the possible combinations of concepts represented by 1 and 0

```
library(dplyr)
# I defined 4 different concepts, therefore, there should be 16 diffe
concept.state <- expand.grid(c(0,1),c(0,1),c(0,1),c(0,1))
# run unique() and check the total of row (should be 16)
concept.state <- unique(concept.state)
paste("the total unique rows should be 16, and the result is .. ", nr
```

```
## [1] "the total unique rows should be 16, and the result is ..  16"
```

```
concept.state <- as.matrix(concept.state)
colnames(concept.state) <- c("c1", "c2", "c3", "c4")
```

Ideal Response Vectors

- Generate the Ideal Response Vectors for each concept state. The Ideal Response Vector would be the pattern of answers we expect a student to give for each concept state.

```
empty.df <- data.frame(matrix( , nrow = 16, ncol = 14))
names(empty.df) <- (c("q1", "q2", "q3", "q4","q5", "q6","q7", "q8","q
DF2 <- data.frame(cbind(concept.state, empty.df))
# plug in IRV for all 14 questions
DF2$q1 <- DF2$c1
DF2$q2 <- DF2$c1
DF2$q3 <- DF2$c1
DF2$q4 <- DF2$c1
DF2$q5 <- 1
DF2$q6 <- DF2$c1
DF2$q7 <- as.numeric(DF2$c1 & DF2$c2)
DF2$q8 <- as.numeric(DF2$c1 & DF2$c2)
DF2$q9 <- as.numeric(DF2$c1 & DF2$c2 & DF2$c3)
DF2$q10 <- as.numeric(DF2$c1 & DF2$c2)
DF2$q11 <- DF2$c1
DF2$q12 <- as.numeric(DF2$c1 & DF2$c2)
DF2$q13 <- as.numeric(DF2$c1 & DF2$c2)
DF2$q14 <- as.numeric(DF2$c1 & DF2$c2)
```

Upload the class answer data & convert data

- Upload the class answer data and convert it to 1s and 0s.
- See the number of unique response vectors in the class data

```
class.answer <- read.csv("formative1-results-DI.csv", header = TRUE)
class.answer[ , 1:2] <- NULL
class.answer <- ifelse(class.answer == "Yes", 1, 0)
class.answer <- unique(class.answer)
paste("there are ", nrow(class.answer), " unique vectors in the class
```

```
## [1] "there are  20  unique vectors in the class data and  16 Ideal Respons
```

Use **Hamming Distance** to compare response vector with the Ideal Response Vectors

- Compare the number of differing elements in each vector
- Use **hamming.distance()**

```
library(e1071)

vector <- c()
for(i in 1:nrow(IRV)){
        row_sum <- sum(apply(class.answer,1,function(x) x!=IRV[i,]))
        vector <- as.vector(c(vector,row_sum))
}

names(vector) <- c(1:16)
min.value <- min(vector)
print(min.value) # min value = 81, occuring in the 12th concept state
```

```
## [1] 81
```

Compute the **error score**

- For each response vector, multiply the lowest Hamming Distance by the number of students that answered with that response vector
- Sum all these values together. This is the error score for your Q-Matrix.

```
# I will loop all students responses into the IRV vector and sum the
IDR <- IRV[12, ]
# find original student response vectors (before applying unique())
student.response.vectors <- read.csv("formative1-results-DI.csv", hea
student.response.vectors[ , 1:2] <- NULL
student.response.vectors <- ifelse(student.response.vectors == "Yes"

total.error.score <- sum(apply(student.response.vectors, 1, function
paste("the total error score of my Q-Matrix is: ", total.error.score)
```

```
## [1] "the total error score of my Q-Matrix is:  115"
```

Summary & Discussion

1. Based on the 4 knowledge concepts I defined, the total error score of my Q-Matrix is 115

2. If we have fewer concept states (i.e knowledge concepts) then the total error score could be lower since there will be fewer combinitions of Ideal Response Vectors (logically speaking)

3. There is a trade-off between **number of knowledge concepts** and **error score**. This also means we can't just focus on decreasing error score but should make sure the number of concepts fit the test questions, as well as, the naming of the concepts should make sense

4. All of these sound like a laborous work and may never have a perfect answer. However, here is an interesting article telling us maybe machine automation can do it better than humans (if we trust it). Machine Beats Experts: Automatic Discovery of Skill Models for Data-Driven Online Course Refinement