Taking Advantage of Scale by Analyzing Frequent Constructed-Response, Code Tracing Wrong Answers

Kristin Stephens-Martinez
UC Berkeley
Berkeley, CA 94720
ksteph@cs.berkeley.edu

An Ju
UC Berkeley
Berkeley, CA 94720
an_ju@berkeley.edu

Krishna Parashar
UC Berkeley
Berkeley, CA 94720
kcparashar@berkeley.edu

Regina Ongowarsito
UC Berkeley
Berkeley, CA 94720
regina.ongowarsito@gmail.com

Nikunj Jain
UC Berkeley
Berkeley, CA 94720
nikunj.jain@berkeley.edu

Sreesha Venkat
UC Berkeley
Berkeley, CA 94720
sreeshavenkat@berkeley.edu

Armando Fox
UC Berkeley
Berkeley, CA 94720
fox@cs.berkeley.edu

ABSTRACT

Constructed-response, code-tracing questions ("What would Python print?") are good formative assessments. Unlike selected-response questions simply marked correct or incorrect, a constructed wrong answer can provide information on a student's particular difficulty. However, constructed-response questions are resource-intensive to grade manually, and machine grading yields only correct/incorrect information. We analyzed incorrect constructed responses from code-tracing questions in an introductory computer science course to investigate whether a small subsample of such responses could provide enough information to make inspecting the subsample worth the effort, and if so, how best to choose this subsample. In addition, we sought to understand what insights into student difficulties could be gained from such an analysis.

We found that ≈5% of the most frequently given wrong answers cover ≈60% of the wrong constructed responses. Inspecting these wrong answers, we found similar misconceptions as those in prior work, additional difficulties not identified in prior work regarding language-specific constructs and data structures, and non-misconception "slips" that cause students to get questions wrong, such as syntax errors, sloppy reading/writing.

Our methodology is much less time-consuming than full manual inspection, yet yields new and durable insight into student difficulties that can be used for several purposes, including expanding a concept inventory, creating summative assessments, and creating effective distractors for selected-response assessments.

ICER'17, August 18–20, 2017, Tacoma, WA, USA.

ISBN 978-1-4503-4968-0/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3105726.3106188

KEYWORDS

constructed-response questions; introductory computer science; education; massive courses; formative assessments; student errors; code-tracing questions

1 INTRODUCTION

The goals of formative assessment are to provide feedback to the student to improve their attainment in cases of error and to inform the teacher as to how to modify or improve pedagogy to address weaknesses in student attainment [1]. In large-enrollment courses, selected-response questions (e.g. multiple choice) are often used as both formative and summative assessment instruments because they can be mechanically graded. However, it is difficult to write selected-response questions whose distractors effectively target common student misunderstandings [13]. Writing constructed-response questions (CRQs), such as filling in blanks, is easier because the teacher does not need to create specific distractors. Additionally, requiring the student to construct a response may also provide richer insight into their level of understanding compared to asking them to identify a correct choice from a list.

However, manually analyzing constructed responses to determine student errors can be prohibitively time-consuming in large-enrollment courses. We considered the wrong constructed responses from code-tracing questions and set about to answer the following research questions:

- **R1**: Can analyzing a small subsample of wrong constructed responses yield information about student difficulties that makes it worth the time investment?
  - **R1.A**: If so, assuming the same questions are used in subsequent course offerings, can the information so gained be applied to future course offerings, further amortizing the time investment?
  - **R1.B**: If so, how should that subsample be chosen and how large must it be?

- **R2**: What insights about student difficulties can be gained from analyzing the subsample?

We address these questions by examining a corpus of 332,829 responses to 92 code-tracing, univalent (having a single correct answer) CRQs. The data comes from responses by 4,068 students in 3 offerings of a large-enrollment introductory computer science (CS) course. We found that a ≈5% subsample of the most frequent wrong answers covers ≈60% of the wrong constructed responses. The frequent wrong answers are consistent in how much they overlap for a given question set and semester pair, but the level of overlap varies between question sets and semester pairs. Therefore frequency should be taken into account when choosing a sample to inspect.

In addition to discovering student difficulties similar to those found in prior work, we also identify new difficulties not reported in prior work such as language-specific constructs and data structures.

After defining terminology and surveying related work, we describe our data set and the emergent coding process we used to analyze it. We then report on how our analysis informs the answers to our research questions. We conclude with a discussion on known and potential uses for our findings.

2 TERMINOLOGY

- *Machine-marked-wrong answer (MMWA)* - A (question, string) pair the automated system marks as incorrect.
- *Wrong answer* - A MMWA that is actually incorrect, as opposed to a false positive marked incorrect by the automated system.
- *Response* - A (student, MMWA) pair; many students may give the same MMWA.
- *Tag* - Human experts' interpretation of a specific student difficulty that could lead to an observed student error.
- *Taggable* - A wrong answer is taggable if at least one tag can be applied to it.

3 RELATED WORK

Misconceptions leading to student programming errors have been intensively studied. Clancy [5] reviews potential causes of programming misconceptions and inappropriate attitudes that interfere with learning programming. Sorva [17] provides an extensive catalog of novice misconceptions about introductory programming content. We find similar misconceptions, but also new misconceptions about data structures and language-specific constructs, which are less studied in prior work. In addition, because we focused on *how* students got wrong answers, we also found student difficulties with syntax and sloppy reading or writing.

While machine-gradable selected-response exercises with carefully-constructed distractors can reveal common student errors, effective distractors are hard to create [13]. This is true especially in introductory programming courses, where instructors' beliefs about student errors have only a weak correlation with the errors students actually make [3]. Others have therefore attempted to extract information about CS students' misunderstandings from various types of CRQs, such as code explanations [10, 15] or code submissions [7, 9, 12, 14], programming process byproducts such as error logs [4, 6], and univalent CRQs as in our work. Univalent CRQs are particularly appealing: like other CRQs, they can reveal useful information about student difficulties without requiring creation of distractors in advance, but unlike other CRQs, they can be easily machine-graded.

The closest work to ours is Sirkiä and Sorva's work analyzing students' missteps when using a visual simulation tool for program tracing [16]. Students use the tool to indicate at each execution step what they expect the code to do, and the tool records every student mistake. The authors identified 200 mistakes each made by at least 10 students, and they analyzed the 26 most popular mistakes to find them to be either a usability-related issue, previously-known conceptual difficulty, or previously-unreported conceptual difficulty. In contrast, our code-tracing questions elicit only a single answer from the student, namely the overall result of running the code. In addition, our *tags*, which code the way(s) students could arrive at a wrong answer, are the equivalent of student mistakes but we allow multiple tags that in combination or separately could cause the wrong answer. Some questions include intermediate `print` statements so we can gain more fine-grained information as the student traces the code. In this regard, both systems encourage students to fix earlier errors before continuing their code-tracing, so we view them as complementary.

Finally, others outside CS have also used the content of wrong answers to understand a student's current knowledge. For basic arithmetic problems, Tatusoka categorized student errors using a two-dimensional Rule Space, with regions of this space representing erroneous rules students use to solve the problems [19, 20]. We instead assign as many tags as necessary to capture the individual or combined student errors that would lead a student to arrive at a wrong answer.

Another way to understand student difficulties is to construct a student model; this is the approach of Repair Theory [2] and systems such as PROUST [8] and MARCEL [18]. Building such models requires enumerating both the student's knowledge and a "bug" list representing ways to mutate that knowledge. While our technique does not result in an explicit student model, it provides insight into common student difficulties without this up-front cost.

4 DATA COLLECTION AND ANALYSIS

Our data comes from three recent offerings of a large-enrollment introductory CS course that teaches programming and the basics of programming abstraction using Python and Scheme. One formative-assessment activity consists of univalent CRQs involving code-tracing: a typical question (Figure 1, left) presents $1-20$ lines of code and asks the student what the interpreter's state will be at various points during execution. A *question set* groups questions about a similar topic, for example, lambda-expressions.

4.1 Data Collection and Preprocessing

The questions are administered through an automatic system running in a terminal window that poses each question and prevents the student from proceeding to the next question until the current one has been answered correctly; unlimited attempts are allowed. Grading is based on completion of the question sets. We record and timestamp every student response; this corpus forms the basis of our dataset.

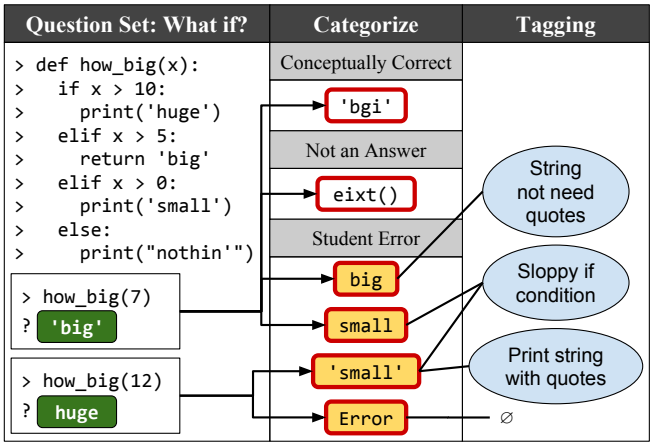


Figure 1: The flow of assigning categories and tags to MMWAs. Left: question set example with two questions on control flow with the correct answers in dark green round-corner rectangles. MMWAs for these questions are in the middle, all with a red bold border. These MMWAs are categorized as: "conceptually correct", "not an answer", or "student error"; the top two categories are false positives. Wrong answers with yellow/non-transparent rounded-corner rectangles are either taggable (top three answers have tags, shown in blue circles) or not taggable (bottom answer).

We cleaned the data by removing all blank answers and any duplicate responses made by the same student. Since the questions are answer-until-correct, every student will have the same set of correct responses, so we examine only their machine-marked wrong responses. Therefore, all future discussion of responses is only about the MMWAs. We did not do any merging of answers, as we found that fixing common typos such as `'TRue'` for `'True'` barely changed our results.

We collected data from the Fall 2015, Spring 2016, and Fall 2016 offerings of the course. Different instructors taught the Fall versus Spring offerings. We report our findings on 11 question sets.

As shown in Figure 1, one weakness of the automatic question administration system is its inflexibility in grading answers, resulting in two kinds of false positives. A *typo* might be the student entering `'bgi'` rather than the correct answer `'big'`; a human instructor would likely recognize that the student was trying to provide the correct answer. A *mode error* might be a student typing `'eixt()'` rather than `'exit()'` when intending to exit a session; these responses are also marked as wrong, even though the student was not attempting to answer the question at all. In the next section, we explain how we deal with such responses in our analysis process.

Figure 2 and Figure 3 summarize information about the question sets we used, ordered chronologically with some order swapping between semesters. Although some question sets showed significant variation in the total number of unique MMWAs across semesters (Figure 2) or percentage of students making at least one mistake when tackling that question set (Figure 3), we find that the properties of the frequent wrong answers remain relatively consistent,

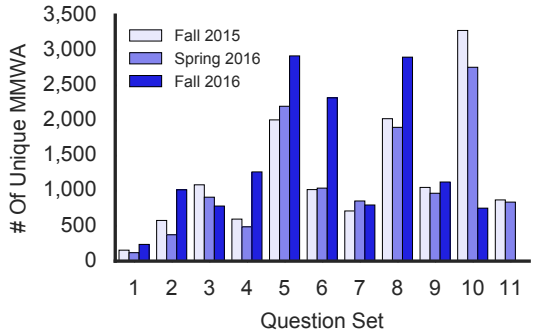


Figure 2: Distribution of the number of unique MMWAs for each semester and each question set. Ex: Question Set 6 had ≈1,000 unique MMWAs for Fall 2015 and Spring 2016 but ≈2,250 for Fall 2016.

as we describe in the next section. A noteworthy outlier is Question Set 1, which has outlier behavior in almost all of our analyses because it tests simple Boolean logic and is therefore easier than the other question sets. In addition, in Fall 2016 that question set was optional, and stronger students (who would be more likely to get all the questions correct on the first attempt) may have simply skipped it.

4.2 Tagging Process

Three content experts inspected the MMWAs. Two were experts who did well in the course and continued on to higher-level courses; the third expert was a former teaching assistant (TA) for the course. For each question set, we first chose a subset of MMWAs. For this set of MMWAs, we completed two phases with multiple steps each.

Our process uses emergent coding to create and assign the tags [11]. Section 5.4 includes details of how much time each phase took, what MMWAs we inspected, inter-rater-reliability, and results of the tagging process.

Phase 1: Tag Creation

(1) Propose Tags: One expert inspects the MMWA set to generate a set of proposed tags with a name, description, and example. (Time here recouped during Phase 2, Step 1 & 2)
(2) Finalize Tags: All three experts discuss the proposed list until deciding on a revised list of final tags.

Phase 2: Tagging Answer Set

(1) Categorize: Two experts separately inspect the MMWA set and assigned each MMWA a category (middle of Figure 1).
- **conceptually correct:** marked wrong due to a typo
- **not an answer:** marked wrong but was not intended as an answer because of a mode error or misunderstanding what text contains the question.
- **student error:** conceptual errors and carelessness, further discussed in a later section.

(2) Assign Tags: Those answers categorized as "student error" are assigned zero or more tags (right of Figure 1).
(3) Consolidate/Discuss: The two experts consolidate their assignments into a single set of category and tag(s), discussing until they reach agreement.

Question Set	Question Count	Fall 2015			Spring 2016			Fall 2016		
		Students	Responses	% Students Wrong	Students	Responses	% Students Wrong	Students	Responses	% Students Wrong
1 Booleans	3	1,271	1,337	54	848	1,046	61	1,093	3,135	83
2 Short Circuit	10	1,283	9,570	95	847	5,234	94	1,692	19,032	99
3 if...else	11	1,293	8,145	97	840	5,843	97	1,022	6,901	99
4 Loops	4	1,278	4,744	81	830	3,766	87	1,675	13,103	96
5 Lambdas	12	1,239	23,124	99	827	19,316	99	1,614	37,666	99
6 HOF	6	1,203	9,698	93	783	8,624	96	1,581	27,121	99
7 OOP	5	907	4,146	92	767	4,147	94	1,516	5,431	90
8 OOP	19	1,042	12,344	99	767	10,760	99	1,510	28,199	99
9 Link Lists	9	1,010	5,543	91	753	4,525	92	1,479	6,478	87
10 Scheme Lists	11	1,040	17,050	99	739	12,791	99	359	3,614	100
11 Iterators	2	870	5,834	92	722	4,562	94	-	-	-

Figure 3: Statistics on the question sets used for this analysis for all course offerings. "HOF" stands for Higher Order Functions and "OOP" for object-oriented programming. "Students" is the number of students attempting that question set; low values are often due to the question sets being optional in certain semesters. "% Students Wrong" is the percentage of students who made at least one error on any question in the question set.

(4) Review/Confirm: During initial training of experts, a third expert inspects consolidated assignments and confirms them. If this expert disagrees, there is a discussion among all three experts until they reach agreement.

5 RESULTS

5.1 R1: Useful to Examine Small Subset of MMWAs?

Our main findings are that frequent MMWAs appear *much more* frequently than infrequent ones, and that for most students, most of their wrong answers are in the frequent set. Therefore, inspecting a small subset of these most frequent MMWAs results in good coverage of cumulative responses covered, rapidly decreasing response coverage, and good coverage of a student's individual MMWAs. This section provides details to support the above findings, which affirmatively answer R1. The results presented refer to the Spring 2016 offering, as there was little difference among the three course offerings. We will note differences as appropriate.

Frequent wrong answers are very frequent. Figure 4 shows the behavior of the 1,000 most frequent MMWAs. Even though Figure 2 shows a wide range in the number of unique MMWAs per question set and per course offering, the cumulative percent of responses covered quickly reaches 50% using no more than the top-100 MMWAs for each question set. In other words, to cover the majority of responses from students, less than 100 MMWAs will need to be inspected per question set, with most question sets needing less than 50. This behavior is confirmed by noticing that the percent of students that submit a given unique MMWA quickly drops to below ≈5% by 100 MMWAs, even though almost all question sets have over 500 unique MMWAs.

For most students, most of their MMWAs are frequent. If we, therefore, use the top-100 unique MMWAs as a simple definition of "frequent MMWAs," we can ask how many students have a given percentage of their MMWAs within that top-100 frequent set. As

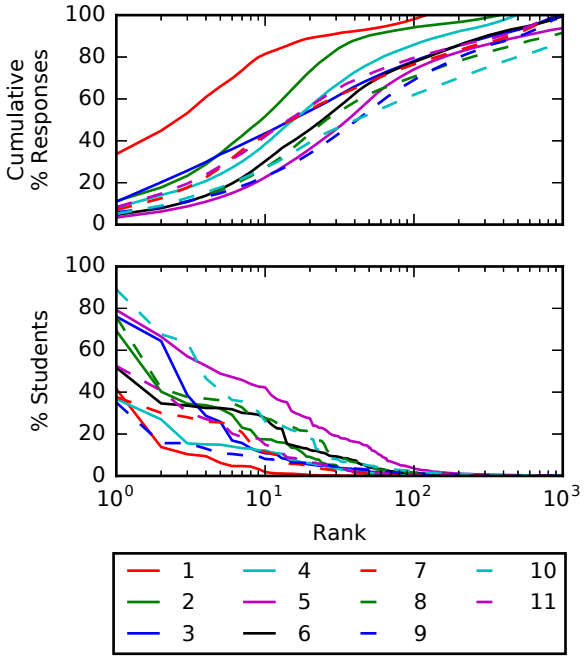


Figure 4: Frequency of the unique MMWAs in Spring 2016. Each line represents a question set. The x-axis (log scale) is the 1,000 most frequently occurring MMWAs ordered by frequency. Upper: Cumulative percent of responses covered by up to the Xth most frequent answer, e.g. Question Set 1's top 10 MMWA covered ≈80% of wrong responses. Lower: Percent of students that submitted the Xth most frequent MMWA, e.g. Question Set 11's 10th most frequent MMWA was submitted by ≈42% of the students that submitted a wrong response to this question set.

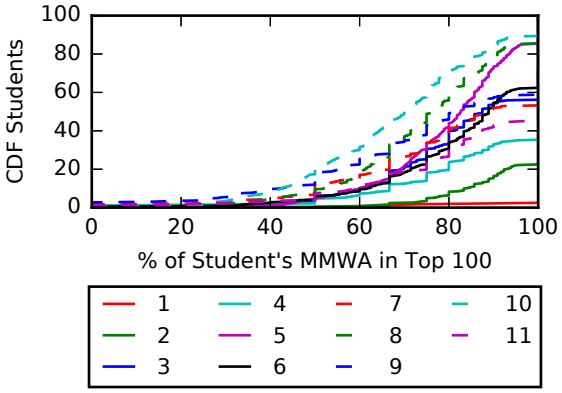


Figure 5: CDF showing the percent of students that have at least the x-axis percentage of their MMWAs in the top 100. Ex: For Question Set 10, ≈75% of students have at least ≈80% of their MMWA in this question set's top 100.

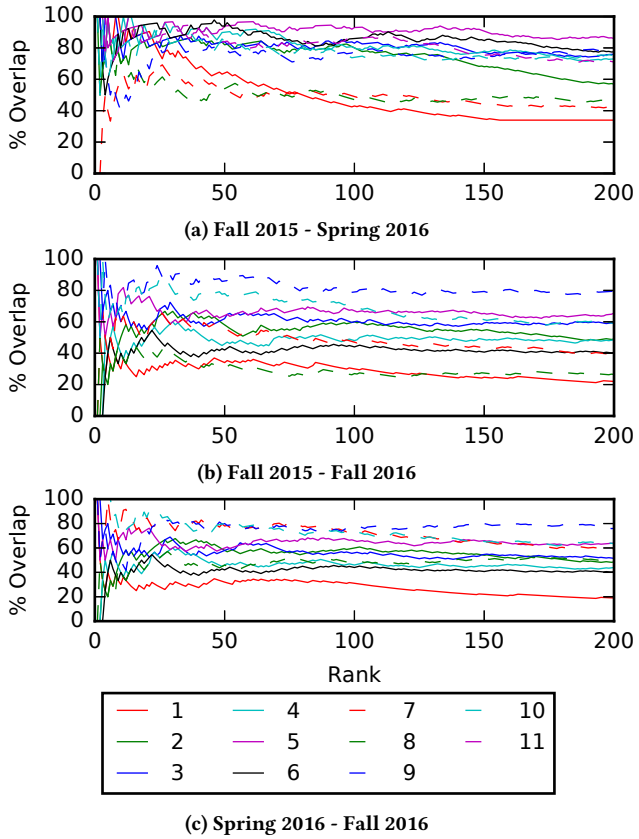
the CDF in Figure 5 shows, for any given question set, 80% or more of the students have the majority of their MMWAs in the top 100. Therefore, most of a student's MMWAs are in the top 100 and the infrequent wrong answers are coming from many students as opposed to a concentrated subset of students. This gives us greater confidence that by inspecting only the most frequent MMWAs, we are examining at least some, if not the majority of, MMWAs from every student.

5.2 R1.A: Are frequent MMWAs stable across course offerings?

Figure 6 shows the overlap of the most frequent MMWAs between a pair of course offerings. (When ordering MMWAs, we broke ties arbitrarily by sorting the answer text alphabetically.) The beginning of the graph is noisy due to a small denominator (the x-axis value). The amount of non-overlapping frequent MMWAs is an estimate of how many MMWAs would need to be inspected in a new offering of the course.

For some question sets there is a high level of overlap between course offerings. Almost all question sets for all pairs of semesters stabilize starting at ≈50 MMWAs and stay stable until ≈150 MMWAs or beyond. 150 is well past the number of MMWAs we need to inspect to cover the majority of responses. Therefore, there will always be some tagging that can be reused, but the amount depends on the question set and other factors that are currently unclear.

The differences between the pairs of semesters is unclear, especially since the Fall 2015 and 2016 offerings were taught by the same instructor, yet have lower MMWA overlap than Fall 2015 with Spring 2016 (since the course material did not change, we would expect that a change of instructor would result in lower overlap than the same instructor teaching the same material twice.). Our best guess as to why this is happening is that the TA staff between the Fall 2015 and Spring 2016 overlapped much more than with the Fall 2016 TA staff. Each course offering had 50 to 87 TAs versus a single lecturer, so it is possible that a teaching effect happening at the TA level is causing the differences in the MMWAs overlap.



(a) Fall 2015 - Spring 2016

(b) Fall 2015 - Fall 2016

(c) Spring 2016 - Fall 2016

Figure 6: Percent of MMWAs that appear in a pair of semester's X most frequent MMWAs, e.g.: comparing Fall 2015 and Spring 2016, ≈85% of Question Set 6's top-100 MMWAs overlapped between the semesters. Note: Fall 2016 did not have Question Set 11 to compare with.

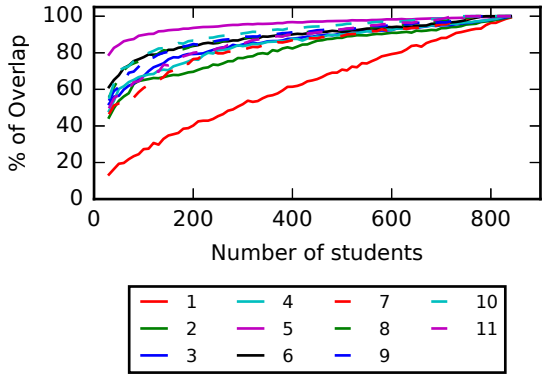


Figure 7: Empirical Monte Carlo analysis results when sampling x-axis students 50 times and plotting the mean overlap across the samples with the entire course offering's top-100 most frequent wrong answers for each question set. Ex: For Question Set 5, when we sampled 100 students 50 times, the mean overlap of the top-100 MMWA with the entire cohort's top-100 MMWA was ≈90%.

Tagging Step	Human effort required
Finalize Tags	≈ 10 mins./tag
Assign Tags	≈ 1.5 mins./answer
Consolidate/Discuss	≈ 0.5 mins./answer
Review/Confirm	≈ 0.1 mins./answer

Figure 8: Human-expert time required for each tagging step. Total time was ≈87 tagger-hours to create tags and ≈36 expert-hours to tag the FrequentSet. We also spent ≈127 expert-hours to tag the StudentSet, required only for our own validation and not integral to the technique.

Figure 7 shows the result of an empirical Monte Carlo simulation on the Spring 2016's data using the following steps and the simple definition of the top-100 unique MMWAs as "frequent": (1) Sample successive values of N students (x-axis value), 50 times, (2) For each sample compute the overlap of the top-100 most frequent MMWAs between the sample and the entire offering, and (3) Plot the mean overlap across the samples.

Figure 7 shows high overlap is achieved between 150 and 250 students, with marginal returns afterward. (Question Set 1 once again is an outlier, most likely due to how few unique MMWAs it had.) Therefore, given our data, frequent MMWAs are stable for a much smaller course than the size we had available (enrollments between 800 and 1,700).

5.3 R1.B: How to choose subsample, and how large?

Using taggability (whether or not a wrong answer has a tag) as a proxy for information about student difficulties, our main finding is that *frequently-occurring wrong answers are more likely to yield information about student difficulties than rarely-occurring ones*, suggesting that the subsample should be created by choosing the most "popular" wrong answers.

We arrived at this conclusion by applying our tagging process to two different MMWA sets. One focused on only the *frequent* MMWAs per question set, hereafter the *FrequentSet*, and the other was *all* MMWAs submitted by a subsample of 50 randomly chosen *students* for each question set, hereafter the *StudentSet*. The main deciding factor for choosing each MMWA set was how much human-expert resources we had to tag these MMWAs. For the FrequentSet, we chose ≈500 MMWAs to tag; for the StudentSet, after getting a better sense of the resources required, we chose ≈2,000. Figure 8 summarizes the person-hours required.

For each question set, we ranked the MMWAs by frequency and then used thresholds on two metrics: (1) total coverage—include the most "popular" MMWAs that cover 60% of all responses, and (2) marginal coverage—then add further MMWAs (still ranked by frequency) as long as each additional MMWA covers at least 0.4% more responses. The interaction between these metrics (Figure 4) led us to explore their value ranges jointly; our parameter values resulted in 508 MMWAs to tag.

Our inter-rater agreement when categorizing wrong answers in the FrequentSet and StudentSet were 96.2% and 86.7% respectively. The fraction of tags given by both experts was 33.2% for the FrequentSet and 46.3% for the StudentSet. The tag overlap is lower than

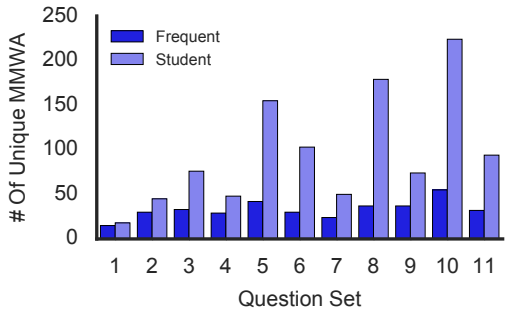


Figure 9: Number of MMWAs in each MMWA set per question set. All MMWAs in the FrequentSet also appeared in the StudentSet and therefore are counted in both bars. Ex: For Question Set 6, the FrequentSet had ≈50 MMWA and the StudentSet had ≈250.

we would like, which could be for three possible reasons that we will address in future work. First, since a question set tested a main topic, a misapplied tag was usually misapplied for multiple answers. Second, in some cases, one expert used a more specific tag than the other, for example, "sloppily evaluating a variable" versus "sloppily evaluating an *attribute* of a variable" (an important distinction in Python). During consolidation, the more specific tag was always used. The inter-rater-reliability scores were not compensated for either of these situations. Third, taggers might need more training. The FrequentSet was the first time tagging for our experts, so training occurred while tagging. The increase in agreement between the FrequentSet and StudentSet, despite there being more wrong answers to tag in the latter, supports this reasoning.

Despite the low agreement on tags, however, the agreement on categories is high, supporting the assertion that frequent wrong answers are more likely to be taggable. In addition, the main focus of this work is on whether the wrong answer was tagged (and therefore whether insights can be gained from it), as opposed to which tag(s) it was given.

By design, there are many more MMWAs in the StudentSet than the FrequentSet (Figure 9). However, the number of MMWAs per question set is much more varied, making this method of choosing MMWAs more likely to result in higher variability in which MMWAs are chosen. In addition, we found the StudentSet included all the MMWAs in the FrequentSet because (in accord with intuition) we were more likely to randomly choose a student that gave a particular frequent MMWA than a particular rare one. Therefore, the graph represents the Frequent MMWAs twice, once in the FrequentSet and once in the StudentSet. This is further evidence that the frequent MMWAs would be sufficient to yield information on student difficulties that is representative of all MMWAs.

Figure 10 also shows that across question sets, there are more MMWAs categorized as "conceptually correct" or "not an answer" for the StudentSet than the FrequentSet. Since our interests are mainly in MMWAs categorized as "student error" and the FrequentSet yields relatively more of these, we have further support that frequency is a good way to choose which MMWAs to inspect.

	Frequent			Student		
	NA	C	SE	NA	C	SE
Mean	1.0%	5.8%	93.1%	4.4%	8.1%	87.5%
Median	0%	4.5%	94.6%	3.2%	4.9%	90.8%
Variance	0.1	0.3	0.4	0.2	0.4	0.4

Figure 10: Statistics on the % of MMWA per category for the FrequentSet and StudentSet. NA - "Not an Answer," C - "Conceptually Correct," and SE - "Student Error."

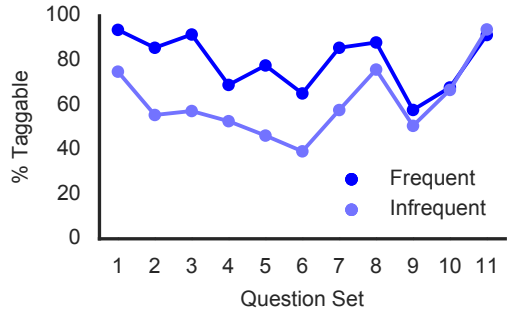


Figure 11: Percent of taggable wrong answers between the frequent and infrequent wrong answers in the StudentSet. Ex: For Question Set 6, ≈67% of the frequent wrong answers in the StudentSet were taggable and 40% of the infrequent wrong answers were taggable. Note: All MMWA in the FrequentSet are the frequent wrong answers in the StudentSet.

Finally, Figure 11 shows that the percentage of taggable MMWAs is higher for the frequent wrong answers than the infrequent ones for all but one question set. While the question sets are ordered chronologically, it is unclear why the percentages are converging but not stabilizing. However, since almost all question sets have a higher percentage of taggable wrong answers for the frequent wrong answers than infrequent wrong answers in the StudentSet, we count this as further evidence that frequent MMWAs are more informative than rare ones.

5.4 R2: What insights can be gained from subsample?

We created a total of 173 tags, which is more than the catalog of novice misconceptions provided by Sorva [17]. In addition, 63% of our tags did not fit the topics in the catalog. This is because our tags focus on what the student did to create the wrong answer, as opposed to the conceptual idea the student misunderstood. In Figure 12 we list the topics from the catalog and ones we created, the number of tags for that topic, and an exemplar tag. Only an exemplar tag is included due to space. A full discussion of the insights we gained will be left to future work.

"Language-Specific Constructs" are tags about student difficulties specific to the language, such as idioms, constructs, and implementations. "Syntax" tags focused on ways students were wrong due to syntax errors or misunderstandings. "Sloppy" tags focused on ways students either read the code they were tracing poorly or submitted their answer without proofreading. Finally, we created the topic "Data Structures" because, even though it is not as well studied in prior work, we found student difficulties with data structures using our analysis. We were able to do this because our data set included questions testing concepts with linked lists, regular lists, dictionaries, sets, and trees.

These insights were gained from question sets created by teaching staff prior to this work and without a rigorous, data-driven design process. We believe more insights could be gained through an iterative process where current insights inform the design of new questions, who's wrong answers are then analyzed, and hopefully more insights are gained. This we, also, leave to future work.

5.5 Research Question Summary

> **R1**: Can analyzing a small subsample of wrong constructed responses yield information about student difficulties that makes it worth the time investment?

Yes, the frequent MMWAs constitute only a ≈5% subsample of the MMWAs in the data set, yet the wrong answers in the FrequentSet are more likely to be taggable.

> **R1.A**: If so, assuming the same questions are used in subsequent course offerings, can the information so gained be applied to future course offerings, further amortizing the time investment?

How much can be applied to another course offering is currently inconclusive. The amount of overlap for a given question set and pair of semesters is consistent between the most frequent ≈50-150 wrong answers. However, how much it overlaps between a question set and pair of semesters is highly dependent on the question set and factors that are currently unclear, such as teaching effects.

> **R1.B**: If so, how should that sample be chosen and how large must it be?

We believe the best way to choose MMWAs is by first ordering them by their frequency and then choosing the most frequent, thresholding based on both the cumulative percent of responses covered and the marginal additional coverage of each additional MMWA. The parameter values can be chosen together based on the number of human-expert hours available for tagging, with the understanding that a lower threshold will affect the results of the set's representativeness and stability.

> **R2**: What insights about student difficulties can be gained from analyzing the subsample?

Using MMWAs from univalent-constructed-response, code-tracing questions with our tagging process, we found both misconceptions similar to those identified in prior work and new misconceptions based on topics appearing in our assessments but not used in prior work, such as difficulties with language-specific constructs and data structures.

6 APPLICATIONS

When we shared our tags with the course's teaching staff, they used the tags to create univalent CRQs for the exams. The wrong answer taggings could also be used to discover common student errors in the class to then create targeted distractors for selected-response assessments and to change the teaching materials to proactively target those errors. In addition, analyzing the MMWAs categorized

Topic	# of Tags	Exemplar Tag		
		Name	Description	Example Code (in Python)

General, Control, OOP, References, Misc	27	Sequential `if` statements are `if...else`	This WA demonstrates that the student believes two if's next to each other are actually an if..else clause	`>>> x = 5`
				`>>> if x <= 5: a = 1`
				`>>> if x > 3: a = 2`
				`>>> print(a)`
				`1`

Variable Assignment	22	Expression not evaluated	This WA demonstrates that the student does not recognize the need to evaluate an expression and instead a code snippet is "passed around."	`>>> a = 1 + 2`
				`>>> a`
				`1 + 2`

Calls	15	Evaluating a function name is a function call	This WA demonstrates that the student believes when the name of a function is in a line of code (but not called) the function is being called.	`>>> f = lambda x: 1`
				`>>> f`
				`1`

Language Specific Constructs*	39	List comprehension does not return a list	This WA demonstrates that the student believes a list comprehension does not return a list, but just a value.	`>>> [x for x in range(3)]` `0`
Syntax*	34	List does not need commas	This WA demonstrates that the student believes a list does not need commas.	`>>> [x for x in range(3)]` `[0 1 2]`
Sloppy*	15	Sloppy sorting	This WA is wrong because the student is being sloppy in how they are sorting the values	`>>> sorted(['b','a','c'])` `['a','c','b']`

Data Structures*	21	Link lists cannot cycle	This WA demonstrates that the student believes that linked lists cannot link back to itself, so if a line of code does that, it is as if did not happen.	`>>> l = Link(1, Link(2, Link(3)))`
				`>>> l.rest = l`
				`>>> l.rest.rest.first`
				`2`

Figure 12: The number of our tags per topic in Sorva's catalog [17] with exemplars. Those with * are topics we created.

as "conceptually correct" can reveal how the system is poor at marking answers correctly. This can lead to either improvements in the automated system or in the questions to reduce such errors. For the MMWAs categorized as "not an answer," they can also be used to understand ways to improve the question or the system. This category led us to discover a confusing question where students thought they were answering the comment that was written as a question next to the code rather than predicting the output of the code. Finally, wrong answer taggings can be used to develop a model to detect student difficulties as they work through the automatically graded assessments. When the model detects a stable difficulty in the student, formative feedback could be delivered immediately in situ.

If the wrong answers tagged in a prior offering of a course do not cover the current offering, more wrong answers need to be collected and tagged. However, the stability we found across cohorts leads us to believe that the desired level of tagged frequent wrong answers will eventually be achieved and/or the number of wrong answers will never be so great as the initial effort of tagging. There is, however, a caveat when using this technique between offerings. If the teacher is using the formative assessments to inform changes in teaching strategies, they are now tracking a moving target since the cohorts are likely changing in their common errors based on the instruction they receive.

7 CONCLUSION

We set out to investigate if the information gained from analyzing responses from univalent-constructed-response, code-tracing questions is worth the effort expended. We analyzed a corpus of 332,829 responses to 92 questions by 4,068 students across 3 offerings of a large-enrollment introductory CS course. We found inspecting the frequent wrong answers are worth the opportunity cost because they are a small sample compared to all the unique wrong answers and cover a majority of the wrong responses. When comparing the overlap of the frequent wrong answers between two course offerings, our results show that the level of overlap is almost always consistent for the frequent wrong answers, but that level varies between question sets and course offering pairs.

In addition, we report on the insights we gained from inspecting these frequent wrong answers. We found similar misconceptions discussed in prior work. Our inspecting process focused on identifying ways students arrive at wrong answers, so we also identified student difficulties with syntax and how students can be sloppy when they read the code or answer questions. Finally, we readily found student difficulties with language-specific constructs (Python and Scheme for this class) and data structures, areas with less prior work on student misconceptions.

REFERENCES

[1] John D Bransford, Ann L Brown, and Rodney R Cocking. 1999. *How people learn: Brain, mind, experience, and school.* National Academy Press.

[2] John Seely Brown and Kurt VanLehn. 1980. Repair theory: A generative theory of bugs in procedural skills. *Cognitive science* 4, 4 (1980), 379–426.

[3] Neil C.C. Brown and Amjad Altadmri. 2014. Investigating Novice Programming Mistakes: Educator Beliefs vs. Student Data. In *Proceedings of the Tenth Annual Conference on International Computing Education Research (ICER '14)*. ACM, New York, NY, USA, 43–50. DOI: https://doi.org/10.1145/2632320.2632343

[4] Adam S. Carter, Christopher D. Hundhausen, and Olusola Adesope. 2015. The Normalized Programming State Model: Predicting Student Performance in Computing Courses Based on Programming Behavior. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15)*. ACM, New York, NY, USA, 141–150. DOI: https://doi.org/10.1145/2787622.2787710

[5] Michael Clancy. 2004. Misconceptions and attitudes that interfere with learning to program. *Computer science education research* (2004), 85–100.

[6] Matthew C. Jadud and Brian Dorn. 2015. Aggregate Compilation Behavior: Findings and Implications from 27,698 Users. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15)*. ACM, New York, NY, USA, 131–139. DOI: https://doi.org/10.1145/2787622.2787718

[7] W Lewis Johnson, Stephen Draper, and Elliot Soloway. 1983. *Classifying Bugs is a Tricky Business.* Technical Report. DTIC Document.

[8] W. L. Johnson and E. Soloway. 1985. PROUST: Knowledge-Based Program Understanding. *IEEE Transactions on Software Engineering* SE-11, 3 (March 1985), 267–275. DOI: https://doi.org/10.1109/TSE.1985.232210

[9] Antti-Jussi Lakanen, Vesa Lappalainen, and Ville Isomöttönen. 2015. Revisiting Rainfall to Explore Exam Questions and Performance on CS1. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15)*. ACM, New York, NY, USA, 40–49. DOI: https://doi.org/10.1145/2828959.2828970

[10] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L. Whalley, and Christine Prasad. 2006. Not Seeing the Forest for the Trees: Novice Programmers and the SOLO Taxonomy. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITICSE '06)*. ACM, New York, NY, USA, 118–122. DOI: https://doi.org/10.1145/1140124.1140157

[11] Matthew B Miles, A Michael Huberman, and Johnny Saldana. 2013. *Qualitative data analysis: A methods sourcebook.* SAGE Publications, Incorporated.

[12] Craig S. Miller and Amber Settle. 2016. Some Trouble with Transparency: An Analysis of Student Errors with Object-oriented Python. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 133–141. DOI: https://doi.org/10.1145/2960310.2960327

[13] Michael C Rodriguez. 2005. Three options are optimal for multiple-choice items: A meta-analysis of 80 years of research. *Educational Measurement: Issues and Practice* 24, 2 (2005), 3–13.

[14] Otto Seppälä, Petri Ihantola, Essi Isohanni, Juha Sorva, and Arto Vihavainen. 2015. Do We Know How Difficult the Rainfall Problem is?. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15)*. ACM, New York, NY, USA, 87–96. DOI: https://doi.org/10.1145/2828959.2828963

[15] Simon and Susan Snowdon. 2014. Multiple-choice vs Free-text Code-explaining Examination Questions. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research (Koli Calling '14)*. ACM, New York, NY, USA, 91–97. DOI: https://doi.org/10.1145/2674683.2674701

[16] Teemu Sirkiä and Juha Sorva. 2012. Exploring Programming Misconceptions: An Analysis of Student Mistakes in Visual Program Simulation Exercises. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research (Koli Calling '12)*. ACM, New York, NY, USA, 19–28. DOI: https://doi.org/10.1145/2401796.2401799

[17] Juha Sorva and others. 2012. *Visual program simulation in introductory programming education.* Aalto University.

[18] James C. Spohrer and Elliot Soloway. 1989. Simulating Student Programmers. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1 (IJCAI'89)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 543–549. http://dl.acm.org/citation.cfm?id=1623755.1623841

[19] Kikumi K Tatsuoka. 1983. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of educational measurement* 20, 4 (1983), 345–354.

[20] Kikumi K Tatsuoka. 1985. A probabilistic model for diagnosing misconceptions by the pattern classification approach. *Journal of Educational and Behavioral Statistics* 10, 1 (1985), 55–73.