ArAl: An Online Tool for Source Code Snapshot Metadata Analysis

Alireza Ahadi
University of Technology
Sydney, Australia
Alireza.Ahadi@uts.edu.au

Raymond Lister
University of Technology
Sydney, Australia
Raymond.Lister@uts.edu.au

Luke Mathieson
University of Technology
Sydney, Australia
Luke.Mathieson@uts.edu.au

ABSTRACT

Several systems that collect data from students' problem solving processes exist. Within computing education research, such data has been used for multiple purposes, ranging from assessing students' problem solving strategies to detecting struggling students. To date, however, the majority of the analysis has been conducted by individual researchers or research groups using case by case methodologies.

Our belief is that with increasing possibilities for data collection from students' learning process, researchers and instructors will benefit from ready-made analysis tools.

In this study, we present ArAl, an online machine learning based platform for analyzing programming source code snapshot data. The benefit of ArAl is two-fold. The computing education researcher can use ArAl to analyze the source code snapshot data collected from their own institute. Also, the website provides a collection of well-documented machine learning and statistics based tools to investigate possible correlation between different variables. The presented web-portal is available at online-analysis-demo.herokuapp.com. This tool could be applied in many different subject areas given appropriate performance data.

CCS CONCEPTS

• **General and reference** → **Metrics**; • **Mathematics of computing** → **Contingency table analysis**; **Exploratory data analysis**; • **Information systems** → **Data mining**;

1 INTRODUCTION

Every year, tens of thousands of students fail introductory programming courses world-wide, and numerous students pass their courses with substandard knowledge. As a consequence, studies are retaken and postponed, careers are reconsidered, and substantial capital is invested into student counseling and support. World-wide, on average one third of students fail their introductory programming course [7]. It has always been of interest to the computing education researcher to identify those who might struggle when learning to program. Particularly, it has been of great interest to identify those who show low performance at early stages of the semester as the novices who fall behind at early stages not only stay behind, but also tend to perform poorly in the final course assessment [4].

Collecting data from students' problem solving processes has led to the creation of several algorithms that can profile students' coding ability [10]. This stream of research became popular beginning with the Error Quotient [11] which is used to quantify how students fix errors in their source code and subsequently to detect students who struggle. This method has been extended to include different parameters such as the error location in the code [14] and time required to fix the error [15].

Adopting such algorithms and approaches outside the original institution can be hard, as their power can partially rely on tacit contextual factors such as tools or how a specific programming language is used [3, 13]. In a broader sense, the computing education research community as a whole struggles to adopt tools such as smart learning content outside the original institutions [8].

Like the authors of the snapshot visualization software SnapViz [6], we see that the problem of adoption can be partially explained through the need to install specific software, and partially through the need to understand and implement the required analysis procedures. To solve both of these specific challenges, we have implemented a (programming) language-independent system called ArAl for the analysis of source code snapshot metadata that can be used, among other things, to provide information on assignments that are highly predictive of course outcomes. By metadata in this context, we mean the number of source code programming snapshots generated by a students when working on a particular programming task.

More specifically, ArAl implements the source code snapshot analysis methodology introduced by Ahadi et al. [1, 5], and provides an easy-to-use online application[1] for the analysis. The system reads in tab-separated data with information on course assignments and exam outcomes, and provides the researcher a list of variables of interest.

While the aforementioned research work on the error quotient and other measures is very promising, in this paper the authors study students in a simpler way, without directly analyzing student code, for the following reasons:

[1]The application is hosted at https://online-analysis-demo.herokuapp.com, and the source code will be made public for publication.

(1) As practicing teachers, we often find ourselves focusing on two simpler aspects of a student's performance in practical programming tasks, before we look at the student's code: (1) whether or not the student provided a piece of code that generates correct answers, and (2) how long it took the student to write the code.
(2) As practicing teachers, we also want ways of assessing the quality of the exercises we are giving the students.
(3) As education researchers, we feel the work on the error quotient and other measures currently lacks a suitable benchmark, a simpler approach, upon which those more complicated approaches are incumbent to improve.

This article is organized as follows. First, we outline the design decisions of the online system and review the analysis methodology. Then, we provide a case study where data from SQL problems of an introductory databases course are analyzed. This section is followed by a second case study where data from an introductory programming course is analyzed to identify students who perform poor in the final exam of the course. This is followed by a discussion on the benefits and limitations of the creation and use of such online tools. Finally, we review the future work and conclude the article.

2 DESCRIPTION OF ARAL

ArAl is an online platform designed for analyzing source code snapshot data. The server side functionality is written in Java, and the front end of the application is written in HTML, JavaScript and CSS. The system runs on a cloud-based server at Heroku[2]. ArAl offers two main features - the first is a descriptive tool where a user can enter the values in the cells of the contingency table manually (representing data for a pair of questions) and observe the output, and the second is where the user can upload a set of data representing multiple questions (in which case ArAl automatically generates all contingency tables and displays those that appear most significant).

The first service module of ArAl is dedicated to the contingency based analysis of two given variables based on their distribution in a 2 by 2 contingency table. This includes analysis of risk factors for unfavorable outcomes, analysis of the effectiveness of a diagnostic criterion for a pre-determined condition such as course outcomes, and measures of inter-rater reliability and measures of association. The second module is concerned with the machine learning based metrics obtained from contingency tables. This includes accuracy, the F1 score (measure of a test's accuracy), the G score (geometric mean), precision, recall, sensitivity, specificity, logical necessity, logical sufficiency etc. All these topics and metrics are presented in great detail in the documentation page of the system.

The second module implements the educational data analytics algorithm presented by Ahadi et al. [5] with an aim of identifying a) students at risk, b) correlation between the performance in the programming task and the pen and paper based assessments, and c) finding reasonable number of steps required to solve the programming task. This module is also based on the contingency tables, but compared to the first modules, it also includes the number of attempts required to complete a programming exercise in the construction of the contingency tables (see Section 2.3).

[2] https://heroku.com

Here, we first discuss the analysis of contingency tables that is the cornerstone of the described system. Then, we outline two of the main features of the system, namely a descriptive tool for understanding contingency tables, and an analytical tool for studying data from individual contexts.

2.1 Analysis of Contingency Tables

The core algorithm of ArAl is dependent on the construction of contingency tables (a variation of the truth table), which are well known statistical constructs. A model of a two by two contingency table is illustrated in Table 1, and in the description here, we assume that the data comes from students. The four variables in Table 1, a, b, c and d represent the number of students who satisfy each of the four possible combinations of the two dichotomous categories, for two given variables V_1 and V_2. The simplest example of a criterion for any two questions V_1 and V_2 is the correctness of students' answers to given questions – for example, a programming assignment in the course and and a question in the exam. Given the example criterion, then a in Table 1 represents students who answered both questions correctly, b represents the students who answered question V_1 correctly but question V_2 incorrectly, c the students who answered question V_1 incorrectly but question V_2 correctly, and d the students who answered both questions incorrectly.

Based on the values of a, b, c and d, one can construct a set of metrics that focus on the association between the two variables (the first module of ArAl). After passing a set of statistical significance tests, the metrics can give useful information on the correlation or the degree of association between given variables.

Note that the contingency tables can be analyzed in a machine learning context where the tables are regarded as a confusion matrices. In this context, the values stored in cells "a", "b", "c" and "d" are regarded as true positive (TP), false negative (FN), false positive (FP) and true negative (TN) respectively.

One of the features of ArAl is the construction of the contingency tables by including the number of steps required/spent on a programming task, regardless of the correctness/score on that task. Ahadi et al. [5] showed that the number of steps, which at a higher level represents the degree of students' engagement with the programming task, can be used as a predictor of success in the course. Hence, ArAl considers the construction of different contingency tables by considering performance, number of attempts, and a combination of both (the second module of ArAl). Contingency tables which review the association between the number of attempts and the score in a given test are mainly concerned with two questions, regardless of the success in the programming task: a) how much effort successful students put on a given programming task, and b) what are the features of the number of attempts spent by unsuccessful students.

One can also use contingency tables that consider both the number of attempts and success ratio in the programming task, and aim to find correlations with the performance in the final exam. These tables can be used to identify upper/lower borders of number of attempts required to successfully complete a programming task, and the correlation between this success and the final exam result. The basic idea is to capture different groups of students [2].

Table 1: A 2 by 2 contingency table constructed based on two variables.

		V_1		
		Category=α	Category=β	row totals
V_2	Category=γ	a	b	r_1
	Category=δ	c	d	r_2
	column totals	c_1	c_2	n

2.2 Feature 1: Descriptive Tool

One of the features of ArAl is the analysis of the association of two given variables that a researcher or an educator can use for study purposes. The tool provides a possibility of entering values to a contingency matrix, and then selecting a metrics of interest that the system should calculate based on the matrix (Figure 1 shows the data entry page). Each metric is explained in detail in the system, and a short list of the metrics analyzed by ArAl is present in Table 2.

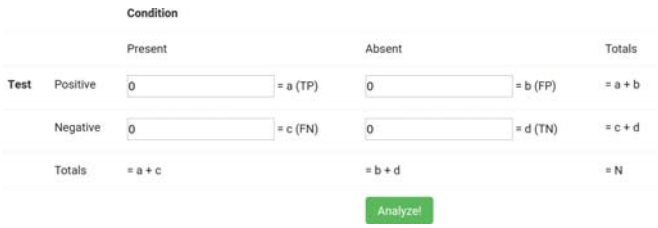

Figure 1: The data entry page for contingency table analysis.

Upon completion of data entry, ArAl checks to make sure that the three following criteria are satisfied: a) no user inserted value for a, b, c or d should be less than 5 – this is related to the functionality of contingency tables, see [16], b) the p value of the x^2 test of the given table is calculated. If the p value is less than 0.01, then the given contingency table is significantly representative of valid associations, regardless of the strength and direction of associations, and c) that $a + b$ is equal to $c + d$, if accuracy is selected, as the positive set size and the negative set size must be equal to calculate some metrics. Upon clicking the calculate button, all metrics are calculated and displayed. Table 2 represents a list of calculated metrics and a short description of what each measures. A detailed explanation of each metric and how they should be interpreted are available on the web-server (See http://online-analysis-demo.herokuapp.com/metrics).

2.3 Feature 2: Analytics Tool

The core feature of ArAl is the analysis of user submitted source code snapshot metadata. The system requires two sets of data files: a) a file including the assessment marks that correspond to the second variable investigated in the contingency table (final exam score, mid-term score, etc.), and b) the source code snapshot metadata with information on the students' attempts for each problem. The former includes the marks and the identifiers of the participants and the latter should include the identifiers, performance on the problem, the identifier of the problem and the number of steps taken by the student on that corresponding problem.

To illustrate the role of the number of attempts in constructing a contingency table, consider 100 students who attempted a programming task. A typical contingency table (as reviewed in the first module) would result in the calculation of the associations between students' score on this programming task (V_1) and their final exam marks (V_2). If we know the number of attempts (consider the number of snapshots generated as the metadata) made by each student, then we can create variations of the original contingency tables. For example, consider the value 8 as the threshold value for the number of attempts. Given this number, we will have 2 different types of students for V_1: those who passed the exercise with less than 8 attempts, and those who failed the exercises with less than 8 attempts. Now, the new contingency table can be constructed based on V_1, and the final exam mark for those students (V_2). Since we know the number of attempts made by each student, we now can define more contingency tables using different values (16, 32, 64, etc). ArAl constructs multiple contingency tables based on the maximum number of attempts observed. ArAl can also generate contingency tables without considering the achieved score in the programming task. As with the previous scenario, using different values for the threshold of the number of attempts, different contingency tables are built where each contingency table considers different types of students according to their number of attempts as V_1 (those with a number of attempts of less than a specific number (8 for example), and those with a number of attempts of more than the specific number) and the score in the final exam (for example) as V_2.

After uploading the files, the user can specify metrics from a provided panel that they want to include in the analysis (See Table 2). After construction of the contingency tables, those tables with significant p-value according to the p-value cut off for the x^2 test specified by the user are displayed. At this stage, a series of rules to filter overlapping tables and identify the most efficient result are applied. The filtering (pruning) rules are as follows:

(1) If the φ values of two contingency tables differ by less than 0.01, then the less general table is pruned.
(2) If the φ values of two contingency tables differ by more than 0.01, and the φ value of the more general contingency table is higher than the φ value of less general table, then the less general table is pruned.
(3) If the φ value of the more general contingency table is lower than the φ value of another table, but a statistical test for significant difference (see [9] and [12]) reveals no significant difference ($p < 0.01$), then the less general bin is pruned.
(4) Contingency tables with accuracy or F1-score of less than 0.5 are pruned.

3 RESULTS

In this Section, we will explore the application of ArAl in two different contexts. In the first case study, we demonstrate how ArAl can be used to identify the possible correlations between the correctness of the answer provided for an SQL question with the correctness of an answer for another question. In the second case study, we demonstrate the correlation between the number of attempts made in coding exercises and the results of the final exam of the course in an introduction to programming course. Note

Table 2: A brief list of quantitative metrics evaluated based on the values extracted from the contingency table.

metric	what does it measure?
odds ratio	the association between an exposure and an outcome
sensitivity	the proportion of positive data points that are correctly categorized
specificity	the proportion of negative data points that are correctly categorized
accuracy	the proportion of correctly categorized data points among all data points
mis-classification rate	the proportion of incorrectly categorized data points among all data points
+ predictive value	proportion of data points categorized as positive that are truly positives
- predictive value	proportion of data points categorized as negative that are truly negatives
false positive ratio	the probability of incorrectly rejecting the true null hypothesis test
false negative ratio	the probability of incorrectly not rejecting a false null hypothesis
false discovery ratio	the rate of type I error in a particular null hypothesis test
F1 score	the harmonic mean of precision and recall
informedness	how informed a predictor is for the specified condition
markedness	how marked a condition is for the specified predictor
φ (also known as MCC)	the association based on adjusting x^2 to factor out sample size
Yule's Q	the intensity of association between two variables
Cramer's V	A measure of association between two nominal variables
diagnostic odds ratio	the effectiveness of a condition
Error Odds Ratio	the standard error for the odds ratio
relative risk	the probability of the occurrence of the event in the presence of a condition to the probability of the event occurrence in the absence of the condition
Kappa	the inter-rater agreement between categorical items.
+ likelihood ratio	the probability of an individual with the condition having a positive test divided by the probability of an individual without the condition having a positive test
- likelihood ratio	the probability of an individual with the condition having a negative test divided by the probability of an individual without the condition having a negative test
difference in proportions	the significance of the difference between two populations based on some single categorical characteristics
relative risk reduction	how much the condition reduced the risk of undesired outcome

that both case studies are focused on the core feature of ArAl (See Section 2.3) .

3.1 Case study 1: Application on SQL Problems

To demonstrate the application of ArAl, we examined inter-relationships between mid-semester SQL test questions in an introductory database course (see *reference removed for review purposes*) for additional details on the data and the context). Data from 2,300 students, with a total of approximately 161,000 attempts at answering SQL queries, were used. Here we investigate whether the performance of students on a question (V_1) predicts whether students will correctly answer another question (V_2) in that same online test. Each question is marked either 0 or 1.

As explained, ArAl can construct the contingency tables based on number of steps required for an exercise, the success score for the exercise or a combination of the two. In our case, we show the output of ArAl with two different types of contingency tables: (1) contingency tables that ignore the success score in completing the

Table 3: The result of the analysis on the source code snapshot meta-data.

Assignment	Correctness	Attempts	Split	φ	F1-score
Robots	1	<3	133/32	0.31	0.77
Robots	1	<7	129/36	0.28	0.75
Robots	1	>127	133/32	0.34	0.77
Robots	1	>31	77/88	0.25	0.64
Loops	1	>15	55/127	0.21	0.5
Loops	1	>31	110/72	0.22	0.67

Table 4: Correlation between different questions V_1 and V_2 when only the number of attempts is the criterion for question V_1.

V_1	V_2	a	b	c	d	attempt	phi
6	7	860	596	272	643	more than 3	.29 (low)
2	4	886	480	400	605	less than 8	.25 (low)
2	7	796	570	336	669	less than 8	.25 (low)
1	4	1027	631	259	454	less than 8	.24 (low)
2	6	539	827	177	828	less than 8	.24 (low)
2	7	999	864	133	375	less than 15	.23 (low)
2	4	1119	744	167	341	less than 15	.22 (low)
1	3	1012	646	271	442	less than 8	.21 (low)
2	3	852	514	431	574	less than 8	.19 (low)
1	5	1098	560	326	387	less than 8	.19 (low)

first SQL question, and (2) contingency tables that ignore the number of steps taken in the students' attempt in completing the SQL question. The aim of both tables is to predict students' performance in a separate SQL question.

3.1.1 Analyzing number of attempts. Table 4 shows the simplified output of ArAl on the correlation analysis between different SQL questions for the contingency tables with the 10 highest φ values. The first two columns of Table 4 (as well as Table 5) describe the type of questions for V_1 and V_2, where 1 represents *SINGLE TABLE QUERIES*, 2 represents *GROUP BY*, 3 represents *GROUP BY WITH HAVING* questions, 4 represents *SIMPLE SUBQUERY*, 5 represents *NATURAL JOIN*, 6 represents *SELF JOIN* and 7 represents *CORRELATED SUBQUERY*. That is, the first row of data in Table 4 has $V_1 = 6$ and $V_2 = 7$, meaning that the system studies the connection between students' performance in writing a self join and a correlated subquery. Columns *a*, *b*, *c* and *d* in Table 4 represent the values from the contingency table used to generate each row of the table. All contingency tables represented in Table 4 have $p < 0.001$ (x^2 test). Table 4 shows that there is only a low correlation between the number of attempts on question V_1 and success in answering question V_2 – that is, the number of attempts that students need to solve a specific SQL question cannot be solely determined by the number of attempts that are needed to solve another SQL question.

3.1.2 Analyzing correctness. Next, we study the same data, but instead of number of attempts, we focus on correctness. That is, the only criterion for question V_1 in Table 5 is whether the student

Table 5: Correlation between different exam questions when the number of attempts for the source question is ignored.

X	Y	a	b	c	d	attempt	phi
4	5	1074	212	350	735	-	.52 (notable)
4	7	903	383	229	856	-	.48 (moderate)
6	7	602	114	530	1125	-	.47 (moderate)
5	7	955	469	177	770	-	.47 (moderate)
3	4	966	317	320	768	-	.45 (moderate)
1	3	1162	570	121	518	-	.42 (moderate)
3	5	1018	265	406	682	-	.42 (moderate)
4	6	620	666	96	989	-	.42 (moderate)
5	6	656	768	60	887	-	.42 (moderate)
2	4	1153	579	133	506	-	.40 (moderate)

answered question V_1 correctly. All contingency tables represented in the rows of Table 5 have $p < 0.001$ (x^2 test). Table 5 shows that there is a moderate correlation between successfully answering given questions in the course.

3.2 Case study 2: Application on Programming source code snapshots

The data for this case study was collected from students enrolled in a 6 week introductory Java programming course at the University of *name removed for review*. In the course, 50% of the overall mark comes from completing online exercises during the 6 week semester, while the other 50% comes from a final exam. Furthermore, to pass the course, students must achieve at least half of the available marks in both the exercises and the final exam.

There are 106 online exercises available for inspection at *link removed for review* but only 77 of those exercises are used in the course. Students were allowed to make multiple submissions for an exercise, henceforth referred to as *attempts*. So in this context, the metadata for a programming task is the number of submissions made to the automated grading system. Attempts could fail for both syntactic and semantic reasons, and the online system provides feedback to students. There were lab sessions where students could seek assistance with the exercises from teaching staff, but students were also allowed to work on and submit the exercises at any other time and place of their choosing. More information about the educational context and the data could be found in *citation removed for review* As data for this paper, each student's performance on each exercise was recorded as two values:

(1) A dichotomous variable; 0 if the student did not succeed in answering the exercise correctly, or 1 if the student did provide a correct answer.
(2) An integer; the number of attempts a student submitted for the exercise (irrespective of whether or not the student provided a correct answer).

As data for this paper, a student's performance on each of those three questions was recorded as a dichotomous variable; 0 if the student's mark for the question was below the class median mark for that question, or 1 if it was above the median.

3.2.1 When number of attempts is ignored. Table 6 shows the exercises with the highest correlation to final exam questions 2, 3 and 4, when the number of attempts by students is ignored. The final exam question numbers are designated in the table's second column, the column headed *ExamQ*.

For final exam question 2, the three highest correlating exercises are 59, 70 and 61, as shown in the third column, headed *Ex*. These three exercises (and all other exercises) are available for inspection at *link removed for review*. The column with the heading *Week* indicates that all the exercises are from weeks 3 and 4 of the 6 week semester.

The columns headed *Q1*, *Q2* and *Q3* show the quartile boundaries for the number of attempts made by students. For example, row 1.1 of the table shows that 25% of the students made 10 attempts or less on the exercise, 50% of the students made 15 attempts or less, and 50% of the students made between 10 and 20 attempts. Note that these quartile boundaries are for all students, irrespective of whether or not they answered the exercise correctly.

The values in the column headed *Correct* indicate that, for the contingency table used to construct each row of this table, answering each exercise correctly is the sole criteria on an exercise.

The columns headed φ and *Accuracy* show the measures of performance as defined earlier. For each exam question, the rows of Table 6 are ordered on φ, from highest to lowest. Both φ and *Accuracy* in each row are calculated using a contingency table as shown in Figure 2. In Table 6, the columns headed *a*, *b*, *c* and *d* show the values for each of these contingency tables. For each row of Table 6, "Meet criterion on exercise x" in Figure 2 is "yes" if a student answered the exercise correctly. For example, column *a* in row 1.1 of Table 6 shows that 32 students answered exercise 59 correctly and also scored above the median mark on final exam question 2. The values recorded for *a*, *b*, *c* and *d* in that row of Table 6 are the four example values shown in Figure 2. The column *sum* is simply the sum of the values in columns *a*, *b*, *c* and *d*, which show that this table used data from approximately 70 students. The sum values in that column vary as sometimes a student did not attempt an exercise. The column headed *p* shows the statistical significance of the contingency table for each exercise, using the standard x^2 test. All exercises shown in this table easily meet the traditional $p < 0.05$ criteria for statistical significance.

		Scored above the class median mark on final exam question Y ?	
		Yes	No
Meets criteria for exercise x ?	Yes	a (e.g. 32)	b (e.g. 6)
	No	c (e.g. 18)	d (e.g. 16)

Figure 2: A 2×2 Contingency Table for Exercise X (V_1) and Final Exam Question Y (V_2): another way of interpreting a contingency table.

3.2.2 When number of attempts is considered. Table 7 shows the exercises with the highest correlation to final exam questions 2, 3 and 4, when the number of attempts made by students on an exercise is considered. Most of the columns in this table contain the same type of information as the previous table. The differences between the previous table and this table are:

- The column headed "Attempts" describes the range of the number of attempts a student must have made on an exercise as part of meeting the criteria on exercise X (as shown in Figure 2). For example, the "≥8" in row 2.1 indicates that a student needed to make at least 8 attempts at the exercise to be counted within either cell *a* or *b* of the contingency table. In rows 2.6 and 2.7, *<16* indicates that a student needed to make less than 16 attempts to meet the criteria.
- Some rows in the column headed *Correct* contain an asterisk. Each of those asterisks indicates that whether a student answered the exercise correctly is irrelevant; the sole criterion for including a student in cell *a* or *b* of the contingency table is the number of attempts the student made on the exercise.
- Rows 2.8 and 2.9 contain *correct* in the column headed *Correct*, indicating there are two criteria that need to be met for a student to be counted in either cell *a* or *b* of the contingency table: (1) the student must have answered the exercise correctly, and (2) the student must have done so in the number of attempts specified in the *Attempts* column.

4 DISCUSSION

The summarized result of the metadata analysis performed by ArAl is reviewed in Tables 4 and 5. The association between the performance in correctly answering different types of questions in the online SQL test is provided in the last column. The φ correlation value when of the contingency tables constructed using the *number of attempts* variable is lower than the contingency tables introduced in Table 5.

That is, in the case of the metadata from the SQL source code snapshots, using the number of attempts taken in answering one SQL question does not strongly relate to the performance in other questions. At the same time, the success in completing a specific question can be a strong determinant of the performance in other questions as well. This result could be interesting from an educational point of view as the degree of positive correlation between different types of questions are not the same. This could be useful (in the sense of having predictive value) for the educator in identifying students who, based on their performance in answering a specific type of SQL question, may perform poorly when answering other types of SQL questions. Also, this sort of information can help the subject coordinators to fine tune the subject curriculum to better meet the course outcome and learning objectives.

As reviewed in the second case study, the metadata (number of attempts) spent on a programming exercise can (with a high prediction accuracy) be a good indicator of performance in the final exam. The researchers can provide ArAl with such information to identify those students who might be in danger of failing the subject or dropping out. The provided data can be collected at any stage of the semester, however, it is better if the data is collected at the early stages of the semester to identify poor performing students as soon as possible.

Table 6: The exercises with the highest correlation to ftnal exam questions 2, 3 and 4, where the sole criterion is whether a student answered the exercise successfully; the number of attempts prior to success is ignored. No exercises correlated signiftcantly ($p < 0.05$) with ftnal exam question 3.

Row No.	ExamQ	Ex	Week	Q1	Q2	Q3	Correct	φ	Acc	a	b	c	d	sum	p
1.1	2	59	3	10	15	20	correct	0.33	0.66	32	6	18	16	72	0.004
1.2	2	70	4	15	40	70	correct	0.32	0.63	29	5	21	17	72	0.005
1.3	2	61	3	11	14	20	correct	0.28	0.61	27	5	23	17	72	0.01
1.4	3	—	—	—	—	—	——	—	—	—	—	—	—	—	——
1.5	4	52	3	16	24	35	correct	0.34	0.66	30	6	18	17	71	0.004
1.6	4	59	3	10	15	20	correct	0.33	0.66	31	7	17	17	72	0.004
1.7	4	55	3	19	26	35	correct	0.31	0.65	30	7	18	17	72	0.007

Table 7: The exercises with the highest correlation to ftnal exam questions 2, 3 and 4, when the number of attempts by students is considered.

Row No.	ExamQ	Ex	Week	Q1	Q2	Q3	Correct	Attempts	φ	Acc	a	b	c	d	sum	p
2.1	2	92	5	4	53	91	*	$\geq$ 8	0.57	0.82	43	7	6	15	71	<0.001
2.2	2	102	6	2	113	199	*	$\geq \geq$ 16	0.48	0.77	38	6	10	15	69	<0.001
2.3	2	103	6	1	86	190	*	$\geq$ 4	0.46	0.76	38	6	11	15	70	<0.001
2.4	3	92	5	4	53	91	*	$\geq \geq$ 4	0.43	0.77	44	10	6	11	71	<0.001
2.5	3	93	5	2	40	69	*	$\geq \geq$ 16	0.36	0.72	38	8	12	13	71	0.002
2.6	3	28	2	7	9	14	*	< 16	0.35	0.75	46	12	6	9	73	0.003
2.7	3	49	3	12	14	18	*	< 16	0.34	0.70	37	8	13	13	71	0.004
2.8	4	52	3	16	24	35	correct	$\geq \geq$ 8	0.38	0.68	30	5	18	18	71	0.001
2.9	4	59	3	10	15	20	correct	$\geq \geq$ 8	0.37	0.68	31	6	17	18	72	0.002
2.10	4	103	6	1	86	190	*	$\geq$ 16	0.37	0.70	34	7	14	15	70	0.002

The increase in systems that collect data from students' problem solving process in various courses has led to a situation where data exists but many do not know what to do with such data. In this article, we have introduced a tool for the analysis of this data, and demonstrated the use of the tool with a data set from a databases course with 2,300 participants.

Our belief is that systems such as ours will be beneficial for researchers and educators, as the system does not require installation or manual coding of the algorithms. Moreover, as the system provides an opportunity to try out the contingency tables manually, it is likely that the users will become familiar with such methodologies. That is, the system can be also used for educational purposes.

Such systems also include downsides. As the system is hosted by a third party, there is an inherent trust issue. What if the system actually stores the data that is sent to it instead of simply analysing it? In our case, the source code of the system will be published, which means that each individual can run or host it at their institution if needed. Another issue is related to the adoption of tools. As research has demonstrated [8], the tools produced by the computing education research community are rarely adopted at other institutions. Will it also be the case for online analytics tools? Perhaps a solution would be to create an online repository or a platform for such tools, to which researchers and educators could refer as needed.

5 LIMITATIONS AND FUTURE WORK

As mentioned earlier, the source code snapshot data and metadata collection by different IDEs has lead to development of multiple algorithms which can measure student's ability in writing/debugging the code. These algorithms can help the computer science educators develop a better understanding of how well their students are doing during the course. However, these algorithms have not been implemented in an online platform and hence the calculation of the metrics introduced by these algorithms is not yet publicly available in an integrated web-based tools. Future work to improve ArAl will be focused on implementation of these metrics. This provides computer science researchers with a platform which makes the use of such metric on their own data possible in an easy fashion. Also, implementation of such metrics within the analysis framework introduced by ArAl makes it possible to investigate the relationship between these metrics and performance in assessments, or other variables of interest.

At the moment, the only variables that can be analyzed are attempts/correctness on a particular assignment compared to outcomes on particular assignments or the final exam. This implementation of ArAl is a first step towards a complete source code snapshot analysis tool.

6 CONCLUSION

In this paper, we have introduced ArAl, a publicly available, open source, online snapshot data analysis tool. The tool mainly offers two streams of service including bi-variate contingency tables based analysis of the association of machine learning bases and statistics based metrics, and the implementation of Ahadi et al's algorithm [5] for source code snapshot data analysis with aims of predicting students' performance, identifying weak students and providing a systematic way to identify the reasonable number of steps required to handle a programming task. Using the data collected from an online SQL test and the data collected from and introduction to programming course, we have demonstrated the application, work pipeline and the correct result interpretation of ArAl.

The contribution of ArAl is three-fold. First it benefits the computing education researcher by providing means of finding the association between computer based assessments and paper based exams. In more general terms, it can be used to identify those assessment items/questions which are more correlated. Secondly, through analysis of the source code snapshot data, ArAl can be used to identify those students who might be in the danger of performing poor in the course. Finally, ArAl can be used to identify interrelationships between the number of steps spent on the programming task, and its the relationship to the course outcome.

REFERENCES

[1] Alireza Ahadi, Arto Hellas, and Raymond Lister. 2017. A contingency table derived method for analyzing course data. *ACM Transactions on Computing Education (TOCE)* 17, 3 (2017), 13.

[2] Alireza Ahadi and Raymond Lister. 2013. Geek Genes, Prior Knowledge, Stumbling Points and Learning Edge Momentum: Parts of the One Elephant?. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13)*. ACM, New York, NY, USA, 123–128. https://doi.org/10.1145/2493394.2493416

[3] Alireza Ahadi, Raymond Lister, Heikki Haapala, and Arto Vihavainen. 2015. Exploring Machine Learning Methods to Automatically Identify Students in Need of Assistance. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15)*. ACM, New York, NY, USA, 121–130. https://doi.org/10.1145/2787622.2787717

[4] Alireza Ahadi, Raymond Lister, and Donna Teague. 2014. Falling behind early and staying behind when learning to program. In *Proceedings of the 25th Psychology of Programming Conference (PPIG '14)*.

[5] Alireza Ahadi, Raymond Lister, and Arto Vihavainen. 2016. On the Number of Attempts Students Made on Some Online Programming Exercises During Semester and their Subsequent Performance on Final Exam Questions. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 218–223.

[6] Evan Balzuweit and Jaime Spacco. 2013. SnapViz: Visualizing Programming Assignment Snapshots. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '13)*. ACM, New York, NY, USA, 350–350. https://doi.org/10.1145/2462476.2465615

[7] Jens Bennedsen and Michael E Caspersen. 2007. Failure rates in introductory programming. *ACM SIGCSE Bulletin* 39, 2 (2007), 32–36.

[8] Peter Brusilovsky, Stephen Edwards, Amruth Kumar, Lauri Malmi, Luciana Benotti, Duane Buck, Petri Ihantola, Rikki Prince, Teemu Sirkiä, Sergey Sosnovsky, Jaime Urquiza, Arto Vihavainen, and Michael Wollowski. 2014. Increasing Adoption of Smart Learning Content for Computer Science Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference (ITiCSE-WGR '14)*. ACM, New York, NY, USA, 31–57. https://doi.org/10.1145/2713609.2713611

[9] Jacob Cohen, Patricia Cohen, Stephen G West, and Leona S Aiken. 1983. Applied multiple regression/correlation for the behavioral sciences. (1983).

[10] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, et al. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports*. ACM, 41–63.

[11] Matthew C Jadud. 2006. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research*. ACM, 73–84.

[12] Athanasios Papoulis. 1990. *Probability and Statistics*. Prentence-Hall International Editions.

[13] Andrew Petersen, Jaime Spacco, and Arto Vihavainen. 2015. An Exploration of Error Quotient in Multiple Contexts. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15)*. ACM, New York, NY, USA, 77–86. https://doi.org/10.1145/2828959.2828966

[14] Maria Mercedes T Rodrigo, Emily Tabanao, Ma Beatriz E Lahoz, and Matthew C Jadud. 2009. Analyzing online protocols to characterize novice Java programmers. *Philippine Journal of Science* 138, 2 (2009), 177–190.

[15] Christopher Watson, Frederick WB Li, and Jamie L Godwin. 2013. Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior. In *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on*. IEEE, 319–323.

[16] F. Yates. 1934. Contingency Tables Involving Small Numbers and the ĬĞ2 Test. *Supplement to the Journal of the Royal Statistical Society* 1, 2 (1934), 217–235.