Students' Behaviours in using Learning Resources in Higher Education: How do behaviours reflect success in Programming Education?

Tai Tan Mai[1], Martin Crane[1], Marija Bezbradica[1]
[1]School of Computing, Dublin City University, Republic of Ireland.

Abstract

Programming education traditionally has been an important part of Information Technology-related degrees but, more recently, it is also becoming essential in many STEM domains as well. Despite this, drop-out rates in programming courses in higher education institutions are considerable and cannot be ignored. At the same time, analysing learning behaviours has been reported to be an effective way to support the improvement of teaching and learning quality. This article aims to deliver an in-depth analysis of students' learning behaviours when using course material items. We analyse an introductory programming course at a University in Dublin. The dataset is extracted from automatically logged learning data from a bespoke online learning system. The analysis makes use of the power of Principal Component Analysis and Random Matrix Theory to reduce dimensionality in, and to extract information from, the data, verifying the results with rigorous statistical tests. Overall, we found that all the students follow a common learning pattern in accessing all given learning items. However, there is a noticeable difference between higher and lower-performing cohorts of students when using practical and theoretical learning items. The high performing students have been consistently active in practice during the study progress. On the other hand, the students who failed the exam have more recorded activities in reading lecture notes and appear to become discouraged and unmotivated from the practical activities, especially in the later stage of the semester.

Keywords: *Programming Education; Learning Behaviours; Principal Component Analysis (PCA); Random Matrix Theory (RMT); Educational Data Mining.*

1. Introduction

Due to the growth in demand for Information Technology (IT)-related job markets, Education in Computer Programming and related domains has received increasing attention. Furthermore, STEM fields (science, technology, engineering, and mathematics) also require essential programming skills and knowledge, making these types of skills an integral part of any STEM sub-discipline (e.g., Artificial Intelligence, Bioinformatics, Statistics etc.). However, despite the necessity of these skills, there have been considerable drop-out rates in introductory programming courses reported from many studies (Kinnunen & Malmi, 2006). In a recent study using data from 161 universities around the world, the failure rate in introductory programming modules has been reported to be 28% on average, with a huge variation from 0% to 91% (Bennedsen & Caspersen, Michael, 2019).

Results from previous research efforts have concluded that learning behaviours tend to be correlated with students' performance in programming education (Carter & Hundhausen, 2017). 'At risk' students will follow this correlation in their learning behaviours (Na & Tasir, 2017). Many researchers have reported that participation frequencies in various learning sessions have positive effects on learning performance (Al-Shabandar et al., 2017). It has also been shown that *practice* is essential for improving students' programming skills and students should be given opportunities to practice and receive constructive feedback (Ben-Ari, 2001; Höök & Eckerdal, 2015). The effect of the diversity of learning styles on learning scores and satisfaction has also been validated, using the data from an online forum (Shaw, 2012). Furthermore, thanks to the development in educational technology, advanced learning systems now enable us to automatically record a large amount of data, such as on interaction, at fine-grained levels, e.g., at the level of mouse and keyboard events. Based on such data, several studies have revealed that there are great variations in accessing online learning tools, which significantly affect student performances (Li & Tsai, 2017; Lust et al., 2012).

On the other hand, novice programming students typically study both theoretical concepts and practical skills through course material items such as lecture notes, lab instructions and exercises. Studies have also reported that students tend to have difficulties in either understanding concepts or in acquiring practical skills (Qian & Lehman, 2017; Whalley et al., 2006). Although much research efforts have been carried out to study the learning behaviours and their relationship with students' outcomes, the learning behaviours in using material items, however, has not been commonly investigated (Li & Tsai, 2017).

In this paper, we focus on the analysis of students' learning behaviours in using course material items between the higher and lower-performing student cohorts. The data are collected from a bespoke online learning system, developed in our computing department, and analysed by a range of techniques including *Principal Component Analysis* (PCA), *Random Matrix Theory* (RMT) and statistical tests.

The rest of the paper is organized as follows. Section 2 briefly describes the context of the study, datasets and methods used for analysis. Section 3 provides details about the results and discussions, followed by the conclusion in Section 4.

2. Research method

2.1. Context of the study and dataset

This research has been carried out based on a dataset representing the usage of *learning material items* of first-year Software Engineering students in a Medium-size Metropolitan University in Dublin. During the course, learning items are provided to the students weekly, including general information, lecture notes, labsheets and programming tasks. Course material items are delivered in the form of web pages on the bespoke online learning system. We formalise the course material items in this context depending on material type (i.e. *General, Lecture, Labsheet and Practice*) combined with the corresponding week, e.g. *Labsheet_1* refers to the labsheet used in week 1. For the general information items, we denote them as *General_0*. Students' interactive events with the items (e.g. mouse clicking or scrolling, highlighting a piece of texts, switching between two items or submitting codes) are logged on the system database. Based on the logged data, it is possible to extract a dataset containing features that indicate the number of user interactive events in every *course material item* for each student. The data have been collected automatically during the learning progress without any manual intervention from either educators or students.

When finishing the exam, students submit their codes to the online system for automatic grading and they receive the results immediately. A submission is considered "correct" if it passes all the test cases pre-defined by the instructors. Every task is given the same mark proportion and the overall mark is given to students after the exam is finished. A student whose overall grade is less than 40/100 is labelled as "*Lower-Performing*", otherwise, that student is considered as "*Higher-Performing*".

2.2. Analysis method

In this paper, the number of data features depends on the number of learning items, which might lead to "the curse of dimensionality", i.e. too many features in the dataset. We, therefore, utilise the power of the *Principal Component Analysis* (PCA) and *Random Matrix Theory* (RMT) to reduce the number of data dimensions, identifying the key components containing essential information. The results are also verified by a set of statistical tests to detect the important underlying factors in the students' learning behaviours. The detail of the analysis method is discussed below.

Let $m \times n$ matrix $\boldsymbol{D}$ be the extracted dataset from the logged data where m refers to the number of students (data rows), n refers to the number of learning items (data columns). Each value

$\boldsymbol{D}_{ij}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, refer to the number of learning events generated when the student *i* was interacting with the learning item *j*. First, we apply Z-score standardisation to the data, which converts $\boldsymbol{D}$ to $\boldsymbol{D}_{\text{normalised}}$. The correlation matrix $\boldsymbol{C}$ can be calculated from $\boldsymbol{D}_{\text{normalised}}$, followed by the spectrum properties of $\boldsymbol{C}$, i.e. empirical eigenvalues $\lambda_{1,2\ldots n}$ where $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and eigenvectors $U^{1,2\ldots n}$. According to RMT (de Prado, 2020), empirical eigenvalues can be compared with the distribution of eigenvalues from the same size random matrix. Particularly, let the thresholds: $\lambda_{\pm} = \sigma^2\left(1 \pm \sqrt{1/Q}\right)^2$ where $\sigma = 1$ due to $\boldsymbol{D}_{\text{normalised}}$ having an unit variance; $\boldsymbol{Q} = (m/n) \geq 1$; and $\lambda_{\pm}$ are the upper/lower bounds of theoretical eigenvalues distribution. Eigenvalues falling outside of the range $[\lambda_-, \lambda_+]$ are assumed to deviate from the expected values of RMT (Laloux et al., 2000). By comparing the distribution of the empirical eigenvalue with the thresholds, we can identify the key eigenvalues containing specific information in students' learning behaviours.

The eigenvalues and eigenvectors of $\boldsymbol{C}$ can be used to form *principal components* of $\boldsymbol{D}$. After analysing and identifying the key eigenvalues λ_k and eigenvectors U^k, it is possible to project the data on U^k, forming *principal components* (PCs) scores (Abdi & Williams, 2010). The eigenvector components corresponding to each PC can be seen as *loadings* of the PC, indicating how much the feature (i.e. the course material item) contributes to the PC. By testing the difference of the scores of principal components between "*Higher*" and "*Lower-Performing*" student cohorts using a *two-sample t-test* (Cressie & Whitford, 1986), we can verify if there is a difference in learning behaviours between the two cohorts.

Inverse Participation Ratio (IPR) is additionally used to assess the contribution of eigenvector components to the corresponding eigenvector, i.e. the contribution of course material items to the PCs. The IPR of the eigenvector U^k is given by $IPR^k = \sum_{l=1}^{n} \left(u_l^k\right)^4$ where u_l^k is a component of the eigenvector U^k. We focus on the value of $1/IPR^k$ which implies the number of eigenvector components significantly contributing to the *PCs*. Eigenvector components can be investigated to observe the common trend in students' behaviours as well as the difference in the behaviours between student cohorts.

3. Results and Discussion

From more than 2.5 million learning events logged in the system database, we extracted a dataset that contains information of 263 students and 37 learning items used in the programming module over the two academic years (2018 and 2019). The students are classified into two cohorts based on their exam results, i.e., "*Higher-performing*" (141 students) and "*Lower-performing*" (122 students). Applying the analysis method discussed above, we have an *m* x *n* data matrix $\boldsymbol{D}$ where $m = 263$; $n = 37$; $\boldsymbol{Q} = 7.1$; $\lambda_+ = 1.98$; $\lambda_- = 0.38$.

3.1. Select the key information part in the dataset

Figure 1 illustrates the probability distribution of empirical eigenvalues extracted from the dataset and the theoretical eigenvalues predicted by RMT. Overall, the majority of empirical eigenvalues (*91.9%*) falls within the range [$\lambda_- = 0.38, \lambda_+ = 1.98$], which are distributed within the black curve in Figure 1. This is in agreement with previous studies which have found that a large proportion of empirical eigenvalues were predicted by RMT (Daly et al., 2008). This finding indicates that there is a measure of randomness in the majority of the eigenvalues. That is to say, these eigenvalues are merely following a random pattern. The remaining eigenvalues, which are higher than the upper limit $\lambda_+ = 1.98$, are outside the noise area and contain key information about the data. i.e. learning behaviours of the students. As the result, the first three components, which correspond to the three largest eigenvalues (i.e., $\lambda_1 = 10.66$, $\lambda_2 = 4.79$, $\lambda_3 = 2.53$), are selected.

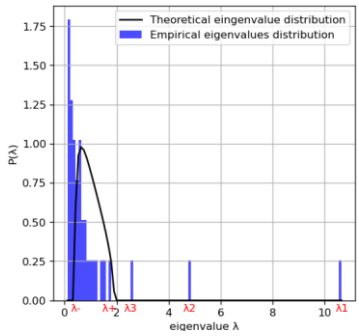

Figure 1. Distribution of empirical eigenvalues and theoretical eigenvalues predicted by RMT

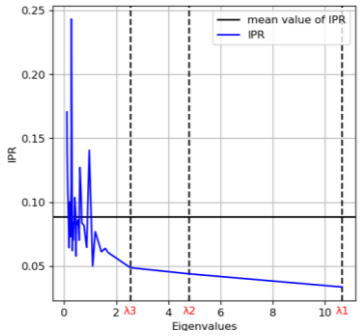

Figure 2. Inverse Participation Ratio of the empirical eigenvalues

Figure 2 shows the IPR values for the eigenvalues of the correlation matrix ***C***. Based on the IPR values of the first three eigenvalues, it is possible to calculate the value $1/IPR$ for λ_1, λ_2 and λ_3. The number of eigenvector components which significantly contribute to the 1st, 2nd and 3rd component are 30, 23 and 23 out of 37, respectively. We note that each eigenvector component refers to a course learning item (e.g., *Labsheet_1*, *Lecture_2* etc.), indicating how much the learning item contributes to the PC. With the high number of learning items contributing to the principal components, the finding implies that the students appear to access and use most of the course material items delivered during the course.

3.3. Principal Component Analysis

Regarding the *first principal component* (PC1), all component loading values are positive. Such a positive value of a component indicates that the component is positively correlated with the corresponding principal component scores. Furthermore, there is no statistical difference between the *PC1* scores for the higher and lower-performing cohorts (*t-test p-value = 0.97 > 0.05*). The statistical testing illustrates the similarity in the learning behaviours

of students in the class. It indicates that almost all students have similar interactions with the course material items during most of the semester. They participated in learning activities and followed the instructions and requirements given by the lecturers. This finding reflects the fact that students were participating in a structured module, i.e., they mostly followed a designated timetable and a similar learning pathway in the class.

To detect dissimilarity between the two cohorts, we created a biplot, a visualisation technique displaying information on both samples and variables in the data matrix (Graffelman, 2014), which represents the 2nd and 3rd principal components (*PC2* and *PC3*) in Figure 3. Each green or red dot refers to a student in the dataset. The blue lines demonstrate the *loadings* of the *PC2* and *PC3*, where each loading line implies how much the corresponding learning item contributes to the *PC2* and *PC3 scores*.

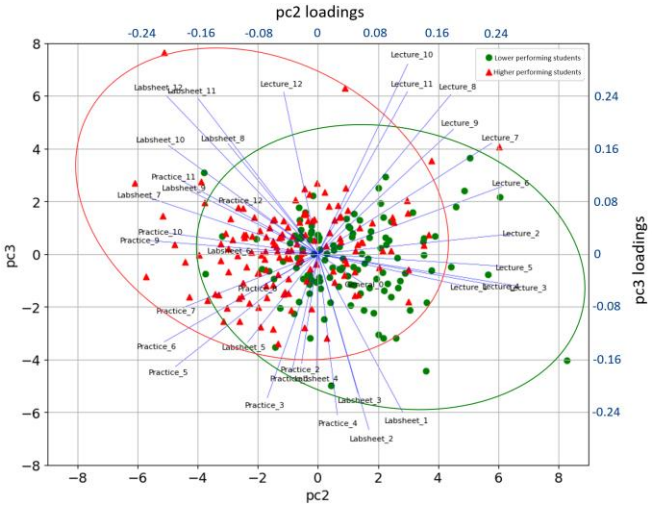

Figure 3. The biplot of the PC2 and PC3 extracted from the dataset.

Both *PC2* and *PC3* comprise either positive or negative component loadings. It is clearly noticeable from Figure 3 that there are clusters of student learning behaviours in the course material items. The majority of the lecture note items are plotted in the top-right quadrant of the graph. Conversely, the practical-related items delivered at the early stage of the study (e.g., labsheets and practices of week 1 to 7) are mostly distributed in the bottom-left quadrant while the remaining practical items which were delivered during the later phases of the course (week 8–12) can be found in the top-left quadrant of the graph. Furthermore, the *PC2* and *PC3* scores for all students are plotted separately from both sides of the two components, bounded by the red and green circles. We understand this finding to indicate the underlying difference in learning behaviours between the two cohorts. The data for the lower-performing cohort (green dots) are plotted on the right side of the graph, which is similar to the loadings of *lecture items*, indicating the positive correlation between them. Meanwhile, the higher-

performing cohort (red dots) has been found to be correlated with the loadings of the *practical* and *labsheet* items. We also notice a statistically significant difference between the two groups in the mean values of the *PC2* and *PC3* scores, with *t-test p-values = 0.00* and *0.007 < 0.05*, respectively. This distinction between the two principal components scores (*PC2 and PC3*) reflects the difference in the learning behaviours between the higher and lower-performing cohorts.

Based on these observations from Figure 3, it can be noticed that higher-performing students appear to pay more attention to the items related to practical activities such as navigating labsheets and doing programming tasks during the learning progress. Practising has been proved to be an effective way to improve programming skills (Ben-Ari, 2001) and higher-performing students have been reported to spend twice as much time on doing programming tasks as the students who failed the exam (Höök & Eckerdal, 2015). Meanwhile, lower-performing students seem to be more active in reading lecture notes than high achieving students. This finding has been in agreement with the conclusions from the previous studies that students who failed the exam spent more time reading the course books than the higher-grade students (Höök & Eckerdal, 2015). A possible reason is that, along with course books, higher-performing students tend to use alternative references from external sources such as consulting senior students and tutorials from the Internet to support their understanding (Rahmat et al., 2012). It is possible to conclude that lower-performing students might face difficulties in understanding new concepts. Therefore, they merely kept reading lecture notes while becoming discouraged and unmotivated to study and practice, especially in the later phase of the semester when the knowledge became more difficult to acquire effectively.

4. Conclusion

In this paper, we conduct an analysis of the learning behaviours of students in using course material items in the context of programming education. The analysis is based on a range of techniques. First, we collected data from a bespoke online learning system. Second, PCA and RMT have been utilised to analyse the data. Using RMT helps to remove random factors and keep the key information part of the dataset. Finally, we analyse principal components scores to find the similarity and difference in students' learning behaviours when they interact with course material items in the course. Overall, students interacted similarly with all course material items during the programming course. However, lower-performing students have been shown to have more activities in reading lecture notes than the higher-performing cohort. Additionally, regarding practical activities, the lower-performing students are found to be more active in practical-related course items including reading lab sheets and solving given programming tasks. Besides, while the higher-performing students consistently practised during the semester, the less achieved students appeared to have lost their focus and motivation in the later phase of the course in solving programming tasks.

There are several possible additions to our approach and research, which will be implemented in future work. The research is currently focusing on the number of interactions in material items. However, other attributes may contain useful information such as time duration spent on the items. Those attributes could enable more insightful analysis of learning processes.

Acknowledgement

This research is financially supported by Irish Research Council.

References

Abdi, H., & Williams, L. J. (2010). Principal component analysis: Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4), 433–459.

Al-Shabandar, R., Hussain, A., Laws, A., Keight, R., Lunn, J., & Radi, N. (2017). Machine learning approaches to predict learning outcomes in Massive open online courses. *2017 International Joint Conference on Neural Networks (IJCNN)*, 713–720.

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45–73.

Bennedsen, J. & Caspersen, Michael. (2019). Failure Rates in Introductory Programming—12 Years Later. *ACM Inroads*, 30–36.

Carter, A. S., & Hundhausen, C. D. (2017). Using Programming Process Data to Detect Differences in Students' Patterns of Programming. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 105–110.

Cressie, N., & Whitford,H. (1986). How to Use the Two Sample t-Test. *Biometrical Journal*, 28(2), 131–148.

Daly, J., Crane, M., & Ruskin, H. J. (2008). Random matrix theory filters in portfolio optimisation: A stability and risk assessment. *Physica A: Statistical Mechanics and Its Applications*, 387(16), 4248–4260.

de Prado, Marcos. (2020). Machine Learning for Asset Managers.

Graffelman, J. (2014). J. Gower, S. Lubbe and N. le Roux, Understanding Biplots, John Wiley and Sons, 2011, pp. 463, *Journal of Classification*, 31(1), 129–133.

Höök, L. J., & Eckerdal, A. (2015). On the Bimodality in an Introductory Programming Course: An Analysis of Student Performance Factors. *2015 International Conference on Learning and Teaching in Computing and Engineering*, 79–86.

Kinnunen, P., & Malmi, L. (2006). Why students drop out CS1 course? *Proceedings of the Second International Workshop on Computing Education Research*, 97–108.

Laloux, L., Cizeau, P., Potters, M., & Bouchaud, J.-P. (2000). Random matrix theory and financial correlations. *International Journal of Theoretical and Applied Finance*, 03(03), 391–397.

Li, L.-Y., & Tsai, C.-C. (2017). Accessing online learning material: Quantitative behavior patterns and their effects on motivation and learning performance. *Computers & Education*, 114, 286–297.

Lust, G., Juarez Collazo, N. A., Elen, J., & Clarebout, G. (2012). Content Management Systems: Enriched learning opportunities for all? *Computers in Human Behavior*, 28(3), 795–808.

Na, K. S., & Tasir, Z. (2017). Identifying at-risk students in online learning by analysing learning behaviour: A systematic review. *2017 IEEE Conference on Big Data and Analytics (ICBDA)*, 118–123.

Qian, Y., & Lehman, J. (2017). Students’ Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Transactions on Computing Education*, 18(1), 1:1-1:24.

Rahmat, M., Shahrani, S., Latih, R., Yatim, N. F. M., Zainal, N., & Rahman, R. A. (2012). Major Problems in Basic Programming that Influence Student Performance. *Procedia - Social and Behavioral Sciences*, 59, 287–296.

Shaw, R.-S. (2012). A study of the relationships among learning styles, participation types, and performance in programming language learning supported by online forums. *Computers & Education*, 58(1), 111–120.

Whalley, J. L., Lister, R., Thompson, E., Clear, T., Robbins, P., Ajith Kumar, P. K., & Prasad, C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies.