UNIVERSITY OF CALIFORNIA, BERKELEY

MIMS FINAL YEAR CAPSTONE

Statistical Analysis of Procedural Learning (Blocked vs. Interleaved) in Humans

Rajvardhan Oak

Mekhola Mukherjee

Vikramank Singh

supervised by

Prof. Zachary Pardos

May 15, 2020

This page was intentionally left blank.

Abstract

Studying student learning patterns, and their outcomes on learning goals is a topic which has been often studied in education related literature. There are two kinds of study patterns which humans employ: blocked and interleaved. In a blocked learning pattern, learners study materials related to the same topic at once, and move on to another topic only when the first one is complete. On the other hand, in interleaved study patterns, this order is scrambled. There have been several studies that analyze the differences in blocked and interleaved study patterns, and comparing learning outcomes in both. However, most of this work has been associated with the study of *declarative learning tasks*, which are memory-based. Therefore, we can argue that such studies look at memory as opposed to learning.

In our work, we focus on *procedural learning*. These are the tasks which you cannot learn by memorization, but have to gradually acquire the skills. We choose computer programming as our declarative task, and study the differences in skills gained between participants in blocked and interleaved patterns. We find that interleaved study pattern works best for the programming task. Furthermore, we also aim to identify the optimal sequence of study modules using statistical modeling techniques. We use the data collected in our study and model it using a Bayesian Network. Our results indicate that it is possible to infer an optimal sequence from performance metrics on several random sequences.

This page was intentionally left blank.

Contents

This page was intentionally left blank.

List of Tables

This page was intentionally left blank.

List of Figures

This page was intentionally left blank.

Chapter 1

Introduction

1.1 Blocked and Interleaved Patterns

Blocked and interleaved settings refer to the way in which learners arrange the various practice/learning exercises by skill. In the blocked setting, tasks are grouped by skill. The learner studies several problems/tasks related to a particular skill, and moves on to the next skill after a sufficient number of problems have been solved. In the interleaved setting, the order of tasks is scrambled; they are not grouped by skill any more.

As a real world example, consider the task of studying mathematics. Assume that the learner wants to study three different skills: Quadratic Equations (Q), Volume of Solids (V) and Trigonometry (T). For every i, let S_i represent a task/problem associated with the skill S. Therefore, $T_1, T_2, T_3 ..... T_n$ will refer to problems related to trigonometry.

In a blocked setting, a learner will choose one of the skills and solve a set of problems related to it. A possible example of blocked pattern would be Q_1, Q_2, Q_3, then T_1, T_2, T_3, and then V_1, V_2, V_3. On the other hand, in the interleaved pattern, one possible sequence could be $T_1, Q_2, T_2, V_3, V_2, T_3, V_1, Q_1, Q_3$. Note that there is no inherent order *within* skill; Q_3 is not any

harder than Q_1, they simply refer to different problems for the skill Q. A visual representation [7] of this concept can be seen in Figure 1.1.

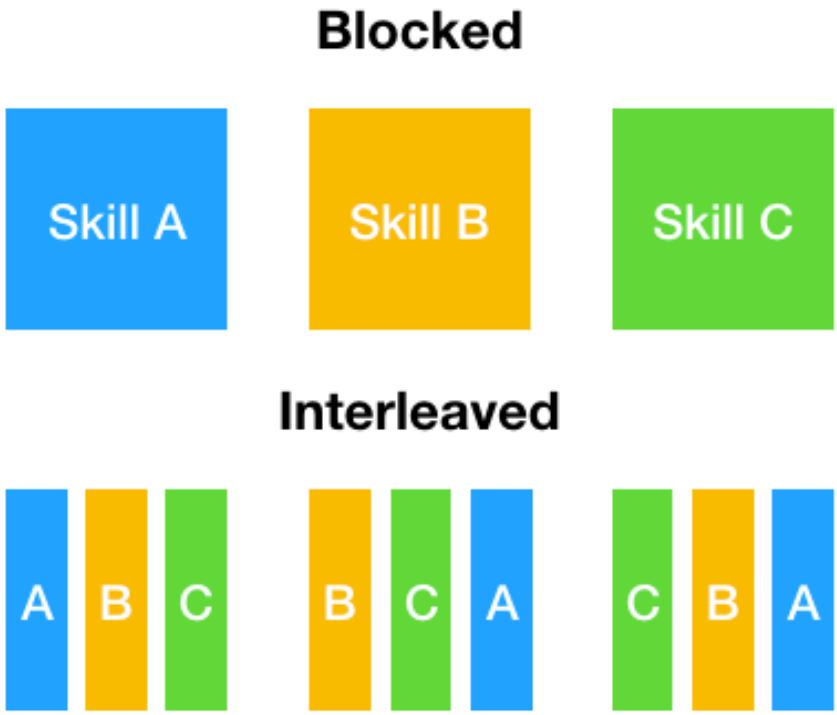

Figure 1.1: Blocked vs. Interleaved Study Patterns [7]

1.2 Procedural and Declarative Tasks

The order in which learners are exposed to various modules and its effect on mastery and long term retention has been widely studied in education-based literature. However, most of the work has been in procedural tasks, not declarative tasks.

Procedural are the tasks which are learnt not by reading or memorizing, but by practice and performance. Typically, they are associated with motor skills [12]. Procedural learning does not depend on fact-remembering. It generally requires repetition of an activity, and associated learning is demonstrated through improved task performance. An example of procedural learning is a child learning to ride a bicycle. One cannot 'remember' how to ride a bicycle; it is only through riding it a few times can they learn it perfectly. Because these tasks cannot be 'remembered', they cannot be 'forgotten' either.

On the other hand, declarative tasks are tasks which can be learnt by memorization. Declarative learning is generally concerned with the learning of facts, events and rules. It talks about learning *what* as opposed to procedural learning which focuses on learning *how* [1]. Declarative learning is about acquisition of facts rather than skills. Typically, declarative knowledge can be talked about, and disseminated to others via discourse. An example of a declarative learning task is learning the capitals of countries, or learning phone numbers. This is something which can be remembered.

1.3 Motivation

As discussed in Section 1.2, most of the work has been on declarative learning. However, it has been proven time and again that procedural skills are more useful for humans. Understanding concepts and being able to apply them to real world problems matters more than memorization of facts. For example, a student must learn not just the technique for solving differential equations, but also use that knowledge to solve a real-world optimization problem. Therefore, it is necessary to understand the differences in learning outcomes for procedural tasks. Doing so will help educators plan and develop better syllabi.

We chose a topic that is close to our heart: computer programming. We wish to evaluate the learning outcomes for coding in blocked and interleaved settings. Note that we do not look at learning the *programming language*, but learning how to solve problems using programming languages. According to IDC, there were $23.2M$ software engineers in the world in 2018 [10]. Therefore, understanding the best manner to study coding, as well as order of concepts to study is of importance.

1.4 Contributions

We aim to answer the following research questions through this study.

- *Is one particular approach (blocked or interleaved) better for learning procedural tasks like computer programming?*

- *Given a set of task sequences and learning outcomes, is it possible to infer a 'best' learning sequence using statistical modeling?*

We summarize our contributions in this work as follows:

- We conduct an exhaustive literature review of the studies and experimental methods that compare learning outcomes in blocked and interleaved patterns.
- We conduct a study on 24 participants and compare the performance between blocked and interleaved settings on a *declarative* task (computer programming). To the best of our knowledge, no other work has studied this task.
- Using a combination of Bayesian Networks and Knowledge Tracing, we identify the optimal sequence of modules to be studied so as to maximize the performance. This shows that given a set of study patterns and their corresponding performances, it is possible to infer the 'best' sequence.

Chapter 2

Review of Literature

2.1 Blocked vs. Interleaved Settings

The differences in learning outcomes in blocked versus interleaved settings have been widely studied. The order in which learners are exposed to various modules and its effect on mastery and long term retention has been of great interest in education-based literature. Most of this experimentation has been in the field of mathematics [16] [20].

Previous research has shown evidence that interleaved learning may show increased performance in mathematical tasks [20]. In [17], college students were tested on finding volumes of cubic solids, and the problems were blocked or interleaved by the type of solid. At a later time, participants were asked to solve previously unseen problems. It was seen that scores in the interleaved group (63%) were almost triple than the blocked group (23%). The study in [3] went a step further and repeated the experiment but at the same time controlling for spaced repetition [22]. The interleaved group still showed significantly better scores as compared to the blocked group (77% vs. 38%). All of these differences were found to be statistically significant within the 95% confidence interval (i.e. $p < 0.05$).

A more recent work [3] shows that the effectiveness of each approach (blocked and interleaved) will depend on what the task at hand demands. In the active learning tasks, where participants are provided feedback after every problem, the interleaved approach was found to be more effective. However, the passive learning tasks had better results with the blocked approach. The difference between the approaches was more starkly seen in the passive study. In another work [19], it was shown that humans tend to prefer blocked approaches, even though the interleaved approach may show better results.

Some work has touched upon interleaving in procedural tasks such as music [23]. In a study, participants with little to no experience in music were exposed to music by different composers and asked to learn or note the music style. Six composers were presented in a blocked pattern, and another six in an interleaved one. It was observed that, in a test over novel music pieces, participants had a better performance in classifying those composers who had been presented in an interleaved manner.

Blocked and interleaved patterns have been studied both at the concept and the module level [24]. In a study where learners were allowed to construct their own learning schedules, it was observed that they chose to block related modules but to interleave topics which were unrelated. After manipulating the sequences as blocked or interleaved within two random groups, it was seen that the optimal schedule must have interleaving at only one of the concept and domain level. No interleaving or interleaving both showed sub-optimal results.

Although the benefits of interleaving have been shown, researchers [3] [20] concede that these have been validated only in the field of mathematics, and may not generalize to other domains. We aim to extend the work and examine its applications in a non-declarative task like coding. To the best of our knowledge, this is the first study that will observe a blocked versus interleaved approach for such a task.

2.2 Procedural vs. Declarative Learning

As seen in Section 1.2, declarative learning is learning *what*, whereas procedural learning is learning *how*. The differences between these two forms of knowledge have been widely studied in the fields of education, psychology and neuroscience. It was shown in an early study that

he underlying memory systems associated in acquiring procedural and declarative knowledge are different from each other [4]. There are physiological differences as well [21]; declarative memory has been known to occupy the medial-temporal region of the brain and the diencephalic system [2]. Procedural memory, on the other hand, has no specific location; it is described as a sequence of actions that draw from various locations of the brain [2].

Procedural and declarative knowledge also play an important role when it comes to competitive, game-based learning. In a study [9] conducted to explore how different knowledge types would influence learners, it was found that learners exposed to procedural knowledge scored lower than those with declarative knowledge on the pre-test, but improved their grades to achieve a comparable level on the post-test. This indicates that the competitve game-based question banks may be an effective learning tool for procedural tasks. A similar study was conducted to examine the differences between procedural and declarative learning in the Assesment of Basic Learning Abilities (ABLA). It was also seen that video aids were more useful in imparting procedural knowledge, and unit testing was more important in declarative learning [8].

2.3 Inferring Optimal Sequences of Study

The task of identifying the optimal sequence for maximizing the knowledge gain has not been studied as deeply as the blocked versus interleaved settings. When it comes to modeling student knowledge as continuously changing during a single session, we apply the Bayesian Knowledge Tracing [5] [25] model. In a BKT model, four parameters are estimated: *forget, learn, guess* and *slip*. BKT can be seen as the modeling of a students state from knowing to not knowing a particular skill. The model assumes each problem to be a learning opportunity, and models the transition probabilities after every problem.

While the basic BKT model assumes that each problem represents an equal opportunities, experiments in [14] examine whether the learning value of an item might be dependent on the particular context the question appears in. Their results show that it was possible to determine statistically significant ordering rules for the questions. The method proposed identifies question ordering rules such as, question X should go before Y in order to obtain the best performance, and are statistically significant. In a similar study [13], it was found that

certain items produce more learning than other given a random sequence. [18] applied the BKT model to the ASSISTMents data [6] and empirically derive better sequences that lead to a higher learning gain. They further conduct a qualitative study into the sequences and find that certain sequences contain a 'desirable difficulty' that introduces some challenges which actually facilitate learning.

As with the blocked and interleaved studies, most of these works have been in online intelligent tutoring systems and hence deal with declarative learning in $K-12$ students. Nowadays, there has been an increase in online tutoring systems for procedural learning such as HackerRank and LeetCode. Studying whether certain sequences lead to better learning is important and relevant for such platforms.

Chapter 3

Experimental Design

3.1 Task Description

The task we chose for this experiment was computer programming. We wanted to test learners' understanding of data structures and algorithms. Note that we do not aim to test the knowledge of the programming language; but the use of that programming language to solve interview-grade coding questions.

We chose three topics/skills which would be practised by our participants: Linked Lists, Binary Trees and Stacks/Queues. Participants were asked to solve two problems related to each topic. Participants were provided a choice of the programming language from C++, Python and Java. Although we expected them to write code, we did accept pseudo-code versions for those who did not have familiarity with those languages. This is in line with the current software engineering interview practices, where getting the logic right is more important than the syntax. Scores were recorded both before (pre-test) and after (post-test).

3.2 Platform

We used the HackerRank [1] online platform to deliver our experiment. HackerRank is an online virtual environment that allows participants to solve programming challenges and practice their coding skills. We chose this platform because of the following salient features:

- **Free.** HackerRank allows the creation and tracking of challenges (or programming concepts) free of charge.

- **Customizable.** The platform allows to create our own challenges, set instructions, assign points, and design custom test cases. It also allows leaderboard and discussion board support. See Figure 3.1.

- **Autograding.** Upon submitting a solution to a problem, HackerRank automatically compiles the code, evaluates it against the test cases and produces a score. It supports compilation for over 20 programming languages. See Figure 3.2 for the coding environment and Figure 3.3 for how the evaluation is provided. This platform facilitates learning because participants can fine-tune their code by looking at the feedback from the test cases.

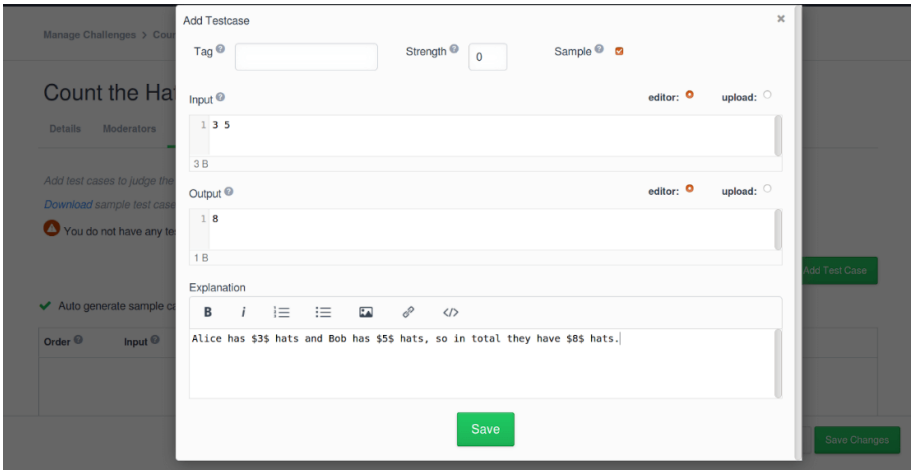

Figure 3.1: Coding Challenge Customization provided by HackerRank

[1] https://www.hackerrank.com/

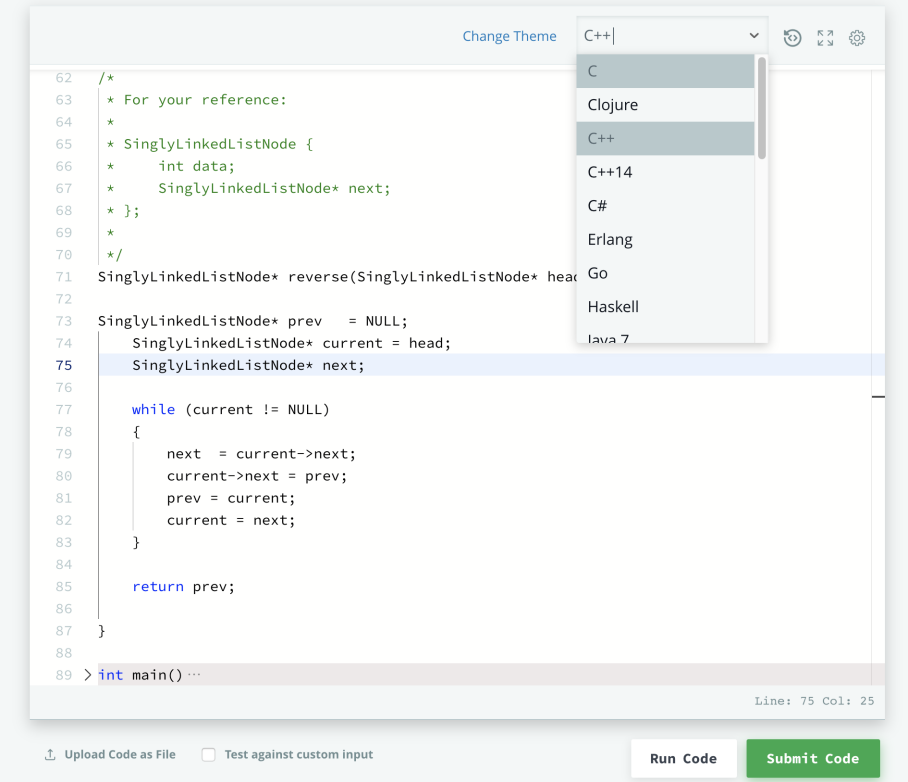

Figure 3.2: Coding Environment provided on HackerRank

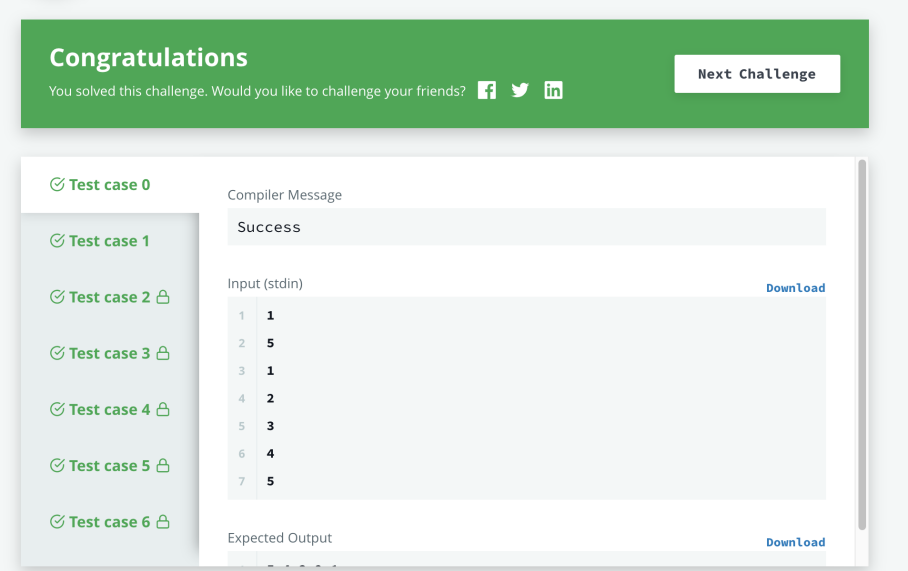

Figure 3.3: Evaluation provided on HackerRank

3.3 Participants

We conducted this study over a sample of $N = 23$ participants. All of the participants were recruited through the personal contacts of the team members. It was difficult to obtain a larger

participation as we were not compensating participants for their time. Out of the 23 participants, 15 were located in the United States and the remaining 8 were from India. All the participants had at least a bachelor's degree. All of them had some familiarity with coding. All participants were contacted via email, and asked to sign up for the contest on the HackerRank platform. Once they had signed up, they were sent further details about the experiment. They were provided a brief overview of the experiment, but no information about the condition they were assigned to.

We collected no demographic or personal data about the participants. The names and email addresses as recorded by HackerRank were not used in our analysis anywhere. We also disabled the leaderboards and discussion forums so that participants would not be able to view each others' scores. In our introductory email, we informed the participants that they could contact us any time to discontinue their participation from the experiment, and to purge their data from our analysis.

3.4 Experiment Design

Our experiment was divided in three phases: Pre-Test, Learning Phase, and Post-Test. We now describe these phases, the experimental conditions, and the questions involved in each phase.

3.4.1 Pre and Post Tests

Because our participants had prior knowledge of programming, it would have been incorrect to draw conclusion from just a single test at the end of the experiment. We conducted a pre and post test, and measured the learning gain by comparing the difference in scores. Both the pre and post test had the same three questions; one each associated with linked lists, trees and stacks/queues. The post-test was delivered in a delayed fashion [15]; participants were shown the post-test 6 days after completing the learning phase. All questions were weighted equally. The questions included in these tests are shown in Table 3.1

Topic	Code	Problem
Linked List	P-1	Given a linked list, determine if there exists a cycle in it.
Binary Trees	P-2	Find the lowest common ancestor of two nodes in a Binary Tree.
Stacks/Queues	P-3	Given a mathematical expression, identify if the arrangement of parentheses is valid.

Table 3.1: Problems in the Pre/Post Test

3.4.2 Learning Phase

3.4.2.1 Tasks

In the learning phase, participants were exposed to 6 coding problems. There were 2 problems related to each of the three topics. We chose one easy and one medium problem. These ratings are assigned by HackerRank based on average performance in those challenges over time. The problems are as shown in Table 3.2.

Topic	Code	Problem
Linked List	LLE	Print the elements of a Linked List in reverse order.
	LLM	Given two linked lists, determine if they merge, and identify the merge point.
Binary Trees	TE	Print the elements of a tree in inorder traversal.
	TM	Given a tree, determine if it is a binary search tree (BST).
Stacks/Queues	SE	Design a function to identify the maximum element in a stack in constant time.
	SM	Given three stacks, find the minimum number of moves needed to equalize their heights.

Table 3.2: Problems in the Learning Phase. The suffixes E and M refer to easy and medium problems respectively

3.4.2.2 Randomization of Participants

Participants were randomly assigned into groups (treatment or control). In the control group, participants were asked to solve the problems in a blocked manner. Note that the topic order was randomized; but problems were always blocked by topic. In the treatment group, participants were exposed to problems in an interleaved fashion. Again, note that the topic-level order was randomized.

In order to create the order assignments, we used a random sequence generator and created 12 blocked and 12 interleaved patterns as shown in Figure 3.4. When participants signed up, we randomly assigned them an ID (from 1 to 24), and sent them relevant instructions.

Candidate	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6	
1	LLE	LLM	HE	HM	TE	TM	
2	HE	HM	LLE	LLM	TE	TM	
3	TE	TM	HE	HM	LLE	LLM	
4	LLE	LLM	TE	TM	HE	HM	
5	TE	TM	LLE	LLM	HE	HM	
6	HE	HM	TE	TM	LLE	LLM	
7	LLE	LLM	HE	HM	TE	TM	BLOCKED
8	HE	HM	LLE	LLM	TE	TM	
9	TE	TM	HE	HM	LLE	LLM	
10	LLE	LLM	TE	TM	HE	HM	
11	TE	TM	LLE	LLM	HE	HM	
12	HE	HM	TE	TM	LLE	LLM	
13	LLE	TE	HE	TM	LLM	HM	
14	HE	LLE	HM	TE	LLM	TM	
15	TE	HE	TM	LLE	HM	LLM	
16	LLE	TE	HE	TM	HM	LLM	
17	TE	LLE	TM	HE	LLM	HM	
18	HE	LLE	TE	HM	TM	LLM	INTERLEAVED
19	TE	LLE	LLM	HE	TM	HM	
20	HE	TE	LLE	HM	TM	LLM	
21	LLE	HE	LLM	TE	HM	TM	
22	TE	HE	LLE	HM	LLM	TM	
23	TE	LLE	TM	HE	LLM	HM	

Figure 3.4: Blocked and Interleaved Problem Sequences

3.4.2.3 Instructions

After a group was assigned, we emailed participants a Google Document containing instructions for the experiment. As HackerRank does not support configuring the order of challenges, we

had to handle the sequencing manually. We created a separate Google Document for every participant, and it contained the order in which they were supposed to solve the tasks.

After the submission of every challenge, we provided participants with a link that contained a sample solution, and associated theoretical explanation. We believe that such feedback after every question helps facilitate learning, and is analogous to real-world scenario where learners refer to solutions in order to do better in future problems.

Participants were asked to abide by the following instructions:

- Complete the experiment by first taking the Pre-Test, followed by the Learning Phase, and finally the Post-Test. They were not expected to complete this in a single session; but had to complete it within a week of starting.
- Strictly abide by the order of problems assigned; and attempt the problems only in the given sequence.
- Attempt the problems to the best of their knowledge only, and not to refer to any material on the Internet, textbooks, online coding forums, or colleagues.
- After every submission, read the feedback provided, and try to understand the solution provided. Do not refer to any other material apart from this.

3.5 Evaluation

HackerRank provides functionality to automatically compile submitted code, and evaluate it against test cases. There are a total of 10 test cases, but participants have access to only 2 test cases. The remaining test cases are hidden from participants to prevent gaming the system by hard-coding the answers. Each question carries 10 points, with all test cases weighted equally. A participant's score in a challenge is directly proportional to the number of test cases passed.

We evaluated participants both before and after the learning phase. We expect that any gain from the learning will be seen in the form of an increase in the scores. Our evaluation metric is the difference between the Pre and Post test scores.

3.6 Critical Assumptions

In conducting this study, we make some underlying assumptions, which, if unfounded, may cast doubt on the validity of our methodology and results. We discuss a few of these assumptions in this section.

First, we depend on participants to abide by the rules, and more importantly, to follow the order of questions assigned to them. Ideally, we wanted to conduct in-person sessions so that we could proctor the learning phase, and make sure that all participants follow the sequence. Unfortunately, due to unavoidable circumstances, in-person sessions were not possible and we risk non-compliance of the treatment; those who were assigned the interleaved treatment may not actually receive it, and those who were not assigned it may receive it.

Second, we assume that the performance in the coding challenges is indicative of the understanding of the concept. This may not always be the case. Some participants may face issues such as not understanding the wording of the question, not being familiar with the programming language, etc. In addition, some participants may have been exposed to the questions beforehand. This would imbibe a prior knowledge that would interfere with their performance in the challenges.

Chapter 4

Analysis and Results

4.1 Preliminaries

4.1.1 Inclusion Criteria

We considered only those subjects in our analysis, who:

- Signed up for the pre-test, post-test and learning phase;
- Completed all the questions (whether successfully or unsuccessfully) in the pre-test and post-test;
- Completed the training phase within 10 days;
- Did not report any anomalous occurrences such as platform malfunction, contact with other participants, or accidental exposure to any material; and
- Did not contact us in order to discontinue participation.

4.1.2 Hypothesis Setting

In standard hypothesis testing, we have a null hypothesis and an alternate hypothesis. The null hypothesis ($\mathscr{H}_0$) states the opposite of what we wish to prove; it states that there is no relation between two groups or variables. The alternate hypothesis ($\mathscr{H}_1$) states that there is a difference. Statistical inference involves either rejecting or accepting the null hypothesis.

Let μ_B and μ_I represent the mean performances in the blocked and interleaved groups respectively. Then we have:

$\mathscr{H}_0$: The means in the two groups are equal; $\mu_B = \mu_I$
$\mathscr{H}_1$: The means in the two groups are unequal; $\mu_B \neq \mu_I$

4.1.3 Analytical Methods

Our metric for comparing the two settings is the average participant score difference in the pre and post-test. We compute these differences and use a *t-test* to evaluate statistical significance. These are statistical tests which are used to determine if there is a significant difference between the mean of two conditions that is unlikely to be due to sampling error or random chance. We use a specific version of the t-test known as an *unpaired t-test* in which we compare the means of two independent groups. The statistic of importance to us is called the p-value. It represents the probability that the obtained mean difference between the two groups simply occurred by chance. For a 95% confidence interval, we typically want $p < 0.05$. T

4.2 Blocked vs. Interleaved Design

Table 4.1 shows the scores obtained by each participant in our study. The Learning Gain is the difference between the pre and post test scores. The group indicates the setting (B denotes blocked and I denotes interleaved).

First, we conducted a t-test to compare the mean post-test scores between the two groups.

ID	Post-Test Scores	Learning Gain	Group
1	22	22	B
2	10	10	B
3	20	10	B
4	30	10	B
5	30	10	B
6	30	30	B
7	30	30	B
8	20	20	B
9	23.3	13.3	B
10	30	30	B
11	30	30	B
12	26	26	B
13	24	24	I
14	10	5	I
15	20	20	I
16	19	19	I
17	20	0	I
18	30	30	I
19	30	10	I
20	22	22	I
21	10	10	I
22	10	10	I
23	15	15	I

Table 4.1: Scores obtained in blocked and interleaved settings

We assume normal distribution pursuant to the Central Limit Theorem. The distribution for both the groups can be seen in Figure 4.1. We found that the average score in the blocked group was 6 points. This difference was statistically significant ($t = 2.11$, $p = 0.04$).

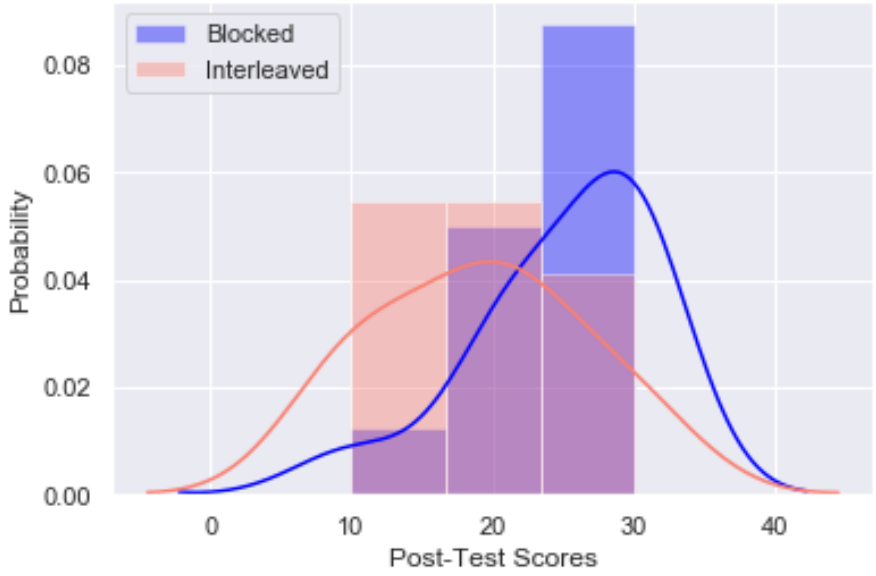

Figure 4.1: Distribution of Post-Test Scores

We then conducted another t-test to compare the mean between the learning gains between the two groups. This is a better measure of the effectiveness of the blocked or interleaved approach, because it also considers prior knowledge as a baseline. The distribution for both the groups can be seen in Figure 4.4. We found that the blocked group was on an average 5 points higher than the interleaved group. However, this difference was not statistically significant ($t = 1.38$, $p = 0.18$).

4.3 Inferring Optimal Sequences

In order to infer the optimal sequence. We first need to define what we mean by an optimal sequence for this case. In our experiment, we are trying to analyse different learning rates for each student. We are also trying to measure the different learning rates for different skills. And then finally come up with the optimal sequence that supports the best learning rate for a student taking into account the learning rate variation for each skill.

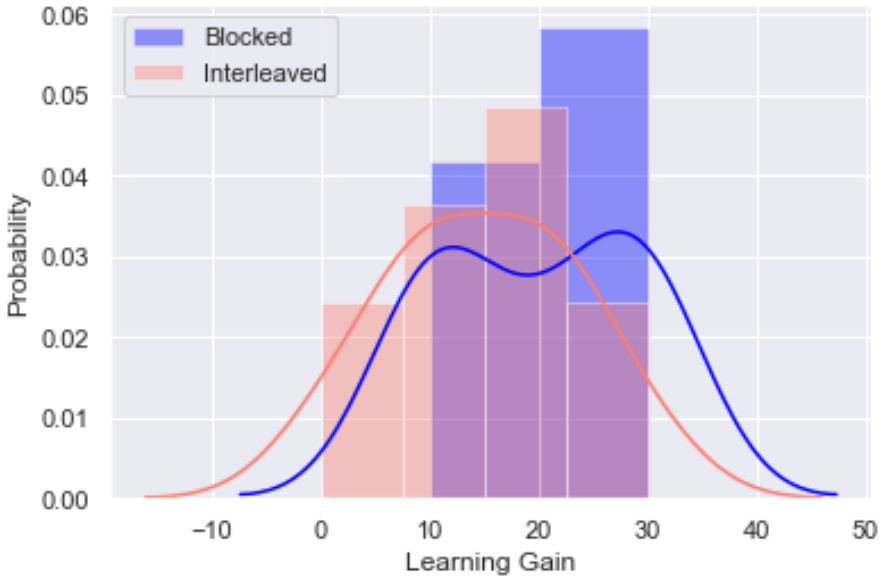

Figure 4.2: Distribution of Learning Gain

From the learning gain scores and the available training sequence of each participant we modelled the latent knowledge for each student. To do this we used Bayesian Knowledge Tracing.

4.4 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) is an algorithm which predicts the latent knowledge of a participant (usually in Intelligent Tutoring Systems) based on a sequence of correct or incorrect answers given by the student [25]. In other words, given a sequence of items or questions that a student has already attempted and solved. Assuming, we have access to the results of those attempts ie. whether the answers were correct or incorrect. BKT is able to predict the probability of the student to get the next question correct.

The BKT model is Hidden Markov model that models the student knowledge as the latent variable and keeps updating this variable based on the answers that a student provides [5]. The assumption here is that the student is able to provide the correct answer to the item or question only after mastering the relevant skill required to apply to the problem.

There are 4 parameters that are tracked by the BKT model and updated as the student

progresses through the given questions [25]. The first paramenter is p-init or p(Lo) which is the probability of the prior knowledge of a student regarding a particular skill. The second parameter is p-transit or p(T) which is the probability that the student will be able to give a correct answer given that they have already encountered a previous state. The third parameter is the p-slip or P(S) which is the probability of a student entering the wrong answer even after knowing the relevant skill. The fourth is p- guess or p(G) which is the probability of a student entering the correct answer even though the student has not yet mastered the relevant skill required to answer that question.

These parameters are updated according to the following rules for one skill for every student [25]:

$$p(L_1)_u^k = p(L_0)^k \tag{4.1}$$

$$p(L_{t+1} \mid obs = correct)_u^k = \frac{p(L_t)_u^k \cdot (1 - p(S^k))}{p(L_t)_u^k \cdot (1 - p(S^k)) + (1 - p(L_t)_u^k) \cdot p(G^k)} \tag{4.2}$$

$$p(L_{t+1} \mid obs = wrong)_u^k = \frac{p(L_t)_u^k \cdot p(S^k)}{p(L_t)_u^k \cdot p(S^k) + (1 - p(L_t)_u^k) \cdot (1 - p(G^k))} \tag{4.3}$$

$$p(L_{t+1})_u^k = p(L_{t+1} \mid obs)_u^k + (1 - p(L_{t+1} \mid obs)_u^k) \cdot p(T)^k \tag{4.4}$$

$$p(C_{t+1})_u^k = p(L_t)_u^k \cdot (1 - p(S^k)) + (1 - p(L_t)_u^k) \cdot p(G^k) \tag{4.5}$$

4.5 Parameter Estimation for BKT

We created the training sequences for each student by encoding the scores of the students to 0 and 1. 0 meant an incorrect response to the question and 1 meant a correct response. For all scores that were below half of the maximum possible score we encoded the answers to be zero. In other words, all scores that were below 5 were marked as incorrect. All scores greater than 5 were marked as correct. Thus, we had a sequence of zeros and ones for each student.

We also calculated the learning gain for each participant by calculating the difference between the pretest and post test scores. We considered this value to be the true learning gain for each student.

We used the pyBKT module [1] to import the functionalities of a BKT model. This module essentially has the BKT and its variations implemented and we used it learn individual students prior, learning rate, guess and slip probabilities. Using PyBKT we learnt the parameters of the BKT Model using Expectation Maximimization for our sequences. We then compared the results of the model with the empirically obtained results.

In our experiment we had 3 different skills that we wanted to measure the BKT parameters for namely Stacks, Trees and Linked List. Our training phase consisted of 6 questions of which there were 2 questions from each topic. We modelled a BKT for each of 3 skills to get knowledge parameters associated to each skill.

We started out with our prior probabilities to be p(T) = 0.30, p(G) = 0.10, p(S) = 0.30 and p(Lo) = 0.15 based on existing literature [11].

4.6 Preparing the data

For a participant in our experiment, an example sequence of question topics looked like this ['A', 'A', 'B', 'B', 'C', 'C']. The same participant's answer sequence looked like this [0,1,0,1,1,1].

In order to differentiate between the different kind of question topic transitions that dif-

[1] https://github.com/CAHLR/pyBKT

ferent students go through, we created a notation of uni-grams and bi-grams. The uni-gram 'A' means that the student encountered the topic 'A' as their first question. The bi-gram 'AB' means that the student encountered question of topic A followed by question of topic B. Including the 3 possible uni-grams and 9 possible bi-grams we had 12 kinds of possible question transitions in our sequence.

We then mapped these transitions onto the questions sequence. A question sequence which earlier looked like 'A', 'A', 'B', 'B', 'C', 'C' now looked like 1, 4, 7, 5, 9, 6 where 1 means the uni-gram 'A', 4 refers to the transition 'AA', 7 refers to the transition AB and so on.

4.7 Learnt parameters

In the learned values for the parameters in parameter estimation through BKT, we can see that most of the learned values do fall in the range of empirical/true parameter values.

We can attribute the large value of Guess to the fact that our questions were in a format (easy, medium, easy, medium) so it might lead to sequences like (0, 1, 0, 1) which is why we believe the guess probability is higher.

4.8 Comparing Blocked and Interleaved Bi-grams

When we compared the whole sequences of blocked and interleaved learning we learnt that blocked performed slightly better than interleaved. We wanted to analyze if we could see the same effect on the unit pairs of blocked and interleaved sequence. We thus compared the average learning rate of blocked bi-grams (eg. AA) and the average learning rates for the interleaved bi-grams (eg. AB).

On comparing all the bi-grams of blocked and interleaved, we found that blocked bi-grams proved to have a higher expected learning of 0.6669148727598269 as compared to the interleaved bi-grams 0.3669382992606846.

```
          truth   learned
prior     0.1500  0.7636
learn1    0.3000  0.1035
learn2    0.3000  0.4577
learn3    0.3000  0.3391
learn4    0.3000  0.0007
learn5    0.3000  1.0000
learn6    0.3000  1.0000
learn7    0.3000  1.0000
learn8    0.3000  0.9995
learn9    0.3000  0.0000
learn10   0.3000  0.0057
learn11   0.3000  0.1964
learn12   0.3000  1.0000
forget1   0.0000  0.0000
forget2   0.0000  0.0000
forget3   0.0000  0.0000
forget4   0.0000  0.0000
forget5   0.0000  0.0000
forget6   0.0000  0.0000
forget7   0.0000  0.0000
forget8   0.0000  0.0000
forget9   0.0000  0.0000
forget10          0.0000  0.0000
forget11          0.0000  0.0000
forget12          0.0000  0.0000
guess1    0.1000  0.0016
slip1     0.3000  0.3912
```

Figure 4.3: Distribution of Learned Parameters

4.9 Computing the Optimal Sequences

We next use the learned probabilities to find the best sequence which has the maximum learning. We need to find the sequence with maximum learning while considering the learning probabilities for all the 3 skills.

We first created an exhaustive list of valid sequences and then calculated the total learning for each such sequence taking into consideration there unique transition probabilities. For the sequences to be valid, we made sure that each such sequence started with a uni-gram which could be either 'A','B' or 'C'. This implies that the student first started with a question of the topic 'A', 'B' or 'C'. The first uni-gram in the sequence was then followed by 5 bi-grams such that the transitions remained consistent. In addition to this, we also ensured that valid sequence

sequences only had 2 questions per skill. An example of a valid sequence 'B', 'BA', 'AC', 'CC', 'CA', 'AB' which is the expanded form of the sequence 'B', 'A', 'C', 'C', 'A', 'B' that the student actually encountered.

We framed this problem as an optimization problem where we want to find a sequences that maximizes the total learning probability but following the above given set of constraints. After calculating the total learning for each of these sequences, we learnt the most optimal sequence with the maximum learning is :

['A', 'B', 'B', 'A', 'C', 'C']

The best 5 optimal sequences were:

['A', 'B', 'B', 'A', 'C', 'C']

['B', 'B', 'A', 'C', 'C', 'A']

['B', 'A', 'C', 'C', 'A', 'B']

['C', 'A', 'B', 'B', 'A', 'C']

['C', 'C', 'A', 'B', 'B', 'A']

4.10 Posterior Inference

To understand how correlated the posterior probability of learning and the actual learning gain calculated by finding the difference between post test and pretest scores we wanted to do a comparison between the posterior knowledge and the learning gains.

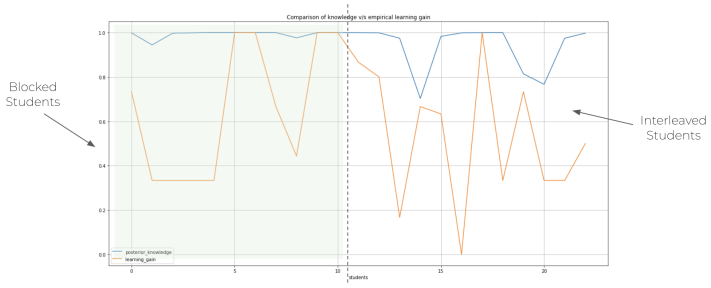

Figure 4.4: Distribution of Learned Parameters

Here we are comparing the post-test scores for each student to their knowledge traced

during the learning phase. We want to understand, Is Post-test score a good representation of actual knowledge gained by the students during the training phase.

On X-axis we have students (0-24) and y-axis is their normalized scores. The blue line represents the posterior knowledge inferred using BKT. The Orange line represents the post-test score for each student.

Students from 0-12 on x-axis belong to the Blocked Group. Students from 12-24 on x-axis belong to the Interleaved Group. We can note from the graph that although the knowledge gained was higher for each student, their performance was not as high on the post-test. However, we do see some correlation between the post-test and latent knowledge for each student. For students in blocked category (0-12 on x-axis), the latent knowledge is higher and so is their post-test performance. For students in interleaved category (12-24 on x-axis), the latent knowledge is relatively lower and fluctuating and so is their post-test scores as compared to the students in blocked category. The high variance in the plots is due to less amount of data and small sequences.

This page was intentionally left blank.

Chapter 5

Summary and Conclusions

Most of the literature in learning theories of education and psychology talks about declarative learning. However, procedural knowledge is more important for understanding the world and applying declarative knowledge to solve problems. We have explored two critical aspects of declarative learning in this work. We chose computer programming as our procedural task because it is an important and a highly relevant one.

We first conducted a study by comparing performances in programming contests between two different groups: one which was exposed to concepts in a blocked manner, and the other in interleaved. We find that the blocked group shows a better performance both in terms of post-test scores as well as learning gain between the pre and post tests. However, only the former effect is statistically significant.

We then explore the inference of the optimal sequence for maximizing learning gain. We apply Bayesian Knowledge Tracing to our problem sequences and find that certain sequences fare better than the others, and that this difference is statistically significant.

We hope that our findings could be applied in online competitive coding platforms such as HackerRank and LeetCode, and help make improvements which can better facilitate optimal learning of coding.

This page was intentionally left blank.

References

[1] M. D. Binder, N. Hirokawa, and U. Windhorst, editors. *Declarative Learning*, pages 931–931. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[2] F. E. Bloom, A. Lazerson, L. Hofstadter, et al. *Brain, mind, and behavior*, volume 300. Freeman New York, 1988.

[3] P. F. Carvalho and R. L. Goldstone. Putting category learning in order: Category structure and temporal arrangement affect the benefit of interleaved over blocked study. *Memory & cognition*, 42(3):481–495, 2014.

[4] N. J. Cohen, H. Eichenbaum, B. S. Deacedo, and S. Corkin. Different memory systems underlying acquisition of procedural and declarative knowledge a. *Annals of the New York Academy of Sciences*, 444(1):54–71, 1985.

[5] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[6] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, 19(3):243–266, 2009.

[7] D. Hendricks. *INCREASE LEARNING RETENTION WITH THESE THREE TECHNIQUES*, 2019 (accessed April 27, 2020).

[8] L. Hu, K. M. Wirth, R. Harris, and J. J. Pear. The evaluation of declarative and procedural training components to teach the assessment of basic learning abilities to senior tutors. *The Psychological Record*, 70(1):163–173, 2020.

[9] G.-H. Hwang, B. Chen, R.-S. Chen, T.-T. Wu, and Y.-L. Lai. Differences between students' learning behaviors and performances of adopting a competitive game-based item bank practice approach for learning procedural and declarative knowledge. *Interactive Learning Environments*, 27(5-6):740–753, 2019.

[10] IDC. *IDC's Worldwide Developer Census, 2018: Part-Time Developers Lead the Expansion of the Global Developer Population*, 2018 (accessed April 29, 2020).

[11] J. Kasurinen and U. Nikula. Estimating programming knowledge with bayesian knowledge tracing. *ACM SIGCSE Bulletin*, 41(3):313–317, 2009.

[12] L. F. Koziol and D. E. Budding. *Procedural Learning*, pages 2694–2696. Springer US, Boston, MA, 2012.

[13] Z. A. Pardos and N. T. Heffernan. Detecting the learning value of items in a randomized problem set. In *AIED*, pages 499–506, 2009.

[14] Z. A. Pardos and N. T. Heffernan. Determining the significance of item order in randomized problem sets. *International Working Group on Educational Data Mining*, 2009.

[15] M. A. Rau, V. Aleven, N. Rummel, and Z. Pardos. How should intelligent tutoring systems sequence multiple graphical representations of fractions? a multi-methods study. *International Journal of Artificial Intelligence in Education*, 24(2):125–161, 2014.

[16] D. Rohrer, R. F. Dedrick, and S. Stershic. Interleaved practice improves mathematics learning. *Journal of Educational Psychology*, 107(3):900, 2015.

[17] D. Rohrer and K. Taylor. The shuffling of mathematics problems improves learning. *Instructional Science*, 35(6):481–498, 2007.

[18] S. Tang, E. McBride, H. Gogel, and Z. A. Pardos. Item ordering effects with qualitative explanations using online adaptive tutoring data. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, pages 313–316, 2015.

[19] S. K. Tauber, J. Dunlosky, K. A. Rawson, C. N. Wahlheim, and L. L. Jacoby. Self-regulated learning of a natural category: Do people interleave or block exemplars during study? *Psychonomic bulletin & review*, 20(2):356–363, 2013.

[20] K. Taylor and D. Rohrer. The effects of interleaved practice. *Applied Cognitive Psychology*, 24(6):837–848, 2010.

[21] T. Ten Berge and R. Van Hezewijk. Procedural and declarative knowledge: An evolutionary perspective. *Theory & Psychology*, 9(5):605–624, 1999.

[22] T. C. Toppino, J. E. Kasserman, and W. A. Mracek. The effect of spacing repetitions on the recognition memory of young children and adults. *Journal of experimental child psychology*, 51(1):123–138, 1991.

[23] S. S. H. Wong, A. C. M. Low, S. H. Kang, and S. W. H. Lim. Learning music composers' styles: To block or to interleave? *Journal of Research in Music Education*, page 0022429420908312, 2020.

[24] V. X. Yan and F. Sana. Does the interleaving effect extend to unrelated concepts? learners' beliefs versus empirical evidence. *Journal of Educational Psychology*, 2020.

[25] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.