30 Days After Introducing Programming: Will My Students Fail?

Márcio Ribeiro[a], Rodrigo Paes[a,*], Rohit Gheyi[b]

[a]*Federal University of Alagoas, Maceió, Brazil*
[b]*Federal University of Campina Grande, Campina Grande, Brazil*

Abstract

The high rate of failing students in introductory programming courses is a common problem. In this context, previous work revealed data and reasons regarding why students fail the courses. Despite these advances, there is a lack of an approach to identify, during the course, that a particular set of students will fail. By having this set, professors and mentors would have time to act and potentially avoid such failings. This way, this paper proposes a strategy to early identify failing students during introductory programming courses. We apply our strategy considering the first 30 days of the course. To evaluate our strategy, we conduct an empirical study regarding 7 courses (3.5 years in total). The study reveals that, from the group of students our strategy points as "likely to fail," 72% of the students indeed fail with 95% standard confidence level. In addition, despite missing 28%, this set seems still interesting, since at least 33% of these students reach the final exam. Therefore, although they pass, we still consider they are good candidates to give special attention as well, which may avoid final exams and lead to better grades.

Keywords: Programming courses, clustering
2010 MSC: 00-01, 99-00

*Corresponding author
Email addresses: `marcio@ic.ufal.br` (Márcio Ribeiro), `rodrigo@ic.ufal.br` (Rodrigo Paes), `rohit@dsc.ufcg.edu.br` (Rohit Gheyi)

1. Introduction

The high rate of failing students in introductory programming courses is a problem []. In particular, this problem has been reported in several universities around the world []. Due to such high failing rates, these courses are often perceived by the students as problematic [1]. In this context, by analyzing several reasons of why the students fail [2], such as lack of motivation, lack of time, and the chosen programming language, previous work report approaches that reduce these rates [1**?**].

These studies focus on data, reasons, and characteristics after the student fail. Nevertheless, there is a lack of an approach capable of identifying that students are not comfortable still during the course. This way, professors and assistants would be able to act and consequently help them. However, identifying potential failing students during introductory programming courses is a non-trivial task [], specially if we consider that this identification must be done as far as possible, otherwise there will be no enough time to act and the student will drop out the course anyway. When considering programming courses, the situation gets worse even when acting just a bit late, due to the strong prerequisites of understanding previous classes to understand the current one (e.g., to understand loops, students must understand conditional structures).

To minimize this problem, in this paper we propose a strategy to *early* identify failing students in introductory programming courses. Our strategy consists of three simple steps. The first one is to make students use a online exercise system. Then, we collect metrics of each student by using such a system. Finally, we execute a clustering algorithm to form groups so that we are able to separate the potential failing students from the other ones. Here, we apply our strategy considering the first 30 days of the programming course.

To evaluate our strategy, we conduct an empirical study regarding 7 courses (3.5 years) with, in total, 227 students. The programming courses have been ministered at the Federal University of Alagoas, Brazil. The results suggest that our strategy can early detect the majority of the failing students within only 30 days. In particular, from the group of students our strategy points as "likely to fail," 72% of the students indeed fail with 95% standard confidence level. Moreover, the 28% our strategy misses is still important to take into account, since at least 33% of these students have difficulties to pass and reach the final exam. So, although they pass, this set has good candidates that also need special attention. In case professors and assistants help these students, they may avoid final exams and achieve better grades at the end of the course.

In summary, this paper provides the following contributions:

- An strategy to *early* identify failing students in introductory programming courses;
- An empirical study assessing the potential of our strategy. We evaluate our strategy by using 227 students from 7 courses during 3.5 years, demonstrating significant potential.

2. Problem

Due to the high number of students, professors and assistants usually do not identify potential failing students during the courses they teach. In this context, professors are not aware of what actually hinders particular students during the learning process. With no additional help, students have no enthusiasm regarding the classes and get frustrated and disappointed, specially when they see their friends excited about the course. This situation causes shame and timidity, which may lead them to not ask questions or not participate in class. When considering introductory programming courses, this frustration commonly lead students to fail the course [].

In this context, there is no easy way to *early* identify potential failing students during introductory programming courses. When professors and assistants do not early identify students that tend to fail the programming course, they may act too late—or even may not act, since there is no available time anymore—and the students drop out the course anyway. In case they act a bit late, they still face the hard task of recovering such students, which happens to be even harder in programming courses, where to understand the next class there is a strong prerequisite of understanding the previous ones (e.g., to understand loops, students must understand conditional structures).

This way, not identifying failing students early is a critical problem, specially when we consider that programming is one of the first courses that students face in their computer science university program. If these courses have as high failure rates as claimed, they could be one of the factors influencing the declining number of students taking a degree in computer science [3].

To minimize the lack of an early identification of potential failing students that would enable professors and assistants to act faster in order to avoid such failings, we next present a strategy that consists of three simple steps: the use of a online judge system, metrics collection, and execution of a clustering algorithm.

3. Early detection of potential failing students

Identifying potential failing students early is very important in the sense that professors still have enough time to act and avoid students to fail the course. In case professors act based on the results of our strategy, we can indirectly help on reducing the high rate of failing students. However, in this paper, we focus exclusively on identifying these students. This way, evaluating and reporting the rate after applying our strategy is outside the scope of this paper.

Next, we detail our strategy to identify these students.

3.1. Online judge system

To assess the students performance during a course, it is important to closely monitor them. In this context, metrics represent an alternative to measure their performance. However, retrieving metrics regarding each student is a difficult and time-consuming task. To minimize this problem, one might rely on online learning tools, since these systems are not only a source for these metrics, but allow their automatic retrieval.

A popular category of this kind of system is the online judges. These systems provide a set of programming problems so that students can submit their solutions. After submitting, the online judge executes the solution against a set of predefined test cases. In case the solution passes over all the tests, the system evaluates the solution as correct. Otherwise, the system evaluates the solution according to the error type (e.g., wrong answer, compilation error, time limit exceeded, runtime error, etc) and even might give important tips so that students can successfully solve the problem.

Since students only learn how to program by programming [4], the online judges consists of an important environment in the sense students can practice and improve their learning process. Also, as the tool is available online, students can find their own rhythms. In addition, online judges play an important role regarding rapid feedback, once professors are often overwhelmed with their daily activities and sometimes are not able to help each student separately.

Given all these advantages, the use an online judge system is the first step of our strategy. In this paper, we use an online judge system named *Huxley* [5]. The system is available online only in portuguese at `http://www.thehuxley.com`. It provides a database composed by more than 300 problems and students are encouraged to use the system, once there is absolutely no penalty in case of wrong solutions. Huxley also classifies the problems according to level of difficulty and programming topics. This allows the students to choose the next problem in accordance to their corresponding levels.

3.2. Metrics

The second step of our strategy consists of metrics retrieval. In particular, we retrieve two metrics by using Huxley. We explain them in what follows:

- **Number of submissions:** this metric represents the number of submissions a student do during the course. As explained, Huxley provides more than 300 problems. To solve a particular one, a student needs to submit at least one correct solution.

- **Number of correct submissions:** if a student solves one problem, we increase this metric by one.

Depending on the level of difficulty of a problem, students may submit several times to solve it. However, notice that this is not necessarily a bad thing regarding the learning process. For example, submitting many times means that students are somehow practicing and studying continuously. Thus, although she is not hitting a right solution, she is trying hard and will eventually hit one. Thus, the number of submissions metric is a good indicator of the amount of practice, which is frequently associated with the level of engaging and learning. However, when considered isolated, a high number of submissions may also mean that the student is getting frustrated due to so many wrong answers she receives. This way, we also consider the number of correct submissions to compensate and help us on studying such cases as well.

[Referencia do porque as metricas sao boas: quanto menos exercicio, maior a probabilidade de levar pau.]

3.3. Clustering algorithm

To identify potential failing students, we use a clustering algorithm to define groups of students. Our idea consists of identifying different groups of students so we can clearly separate students susceptible to fail from the other ones. To compute the groups, the algorithm takes into account the metrics we present in Section 3.2. To make the strategy parameterizable in terms of number of groups, we use the well-known clustering algorithm k-means [], which takes such a number as input.

Notice that the number of groups plays an important role on identifying potential failing students. In this paper, we set the algorithm to compute two and three groups and evaluate both cases. For two groups, we have students who will either fail or pass. For three groups, we have fail, pass, and students in which the strategy will not conclude anything about them, which we name "inconclusive." This is reasonable since our study focuses on the very first

30 days, which means our drawings regarding the inconclusive group might be completely wrong: these students can either improve themselves and pass the course or fail due to several reasons.

We focus on two and three groups basically for two reasons: (i) using one group is useless; and (ii) more than three only brings more inconclusive groups to the table which, given our context, is pretty much the same of having only one inconclusive group.

3.4. Summary

Figure 1 combines all steps of our strategy. As the classes are happening, we encourage students to solve the problems available at Huxley. Although solving exercises is not mandatory, there are some particular activities where the professor forces students to use Huxley, such as formal exams. Next, we collect the metrics we detail in Section 3.2. To identify potential failing students *early*, we collect the metrics for the first 30 days of the introductory programming courses. Then, we execute the k-means clustering algorithm.

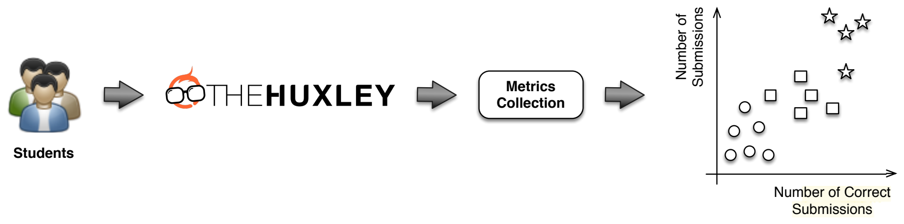


Figure 1: Summary of our strategy to identify potential failing students.

4. Evaluation

To evaluate our strategy, in this section we present the empirical study we conduct. The objective of this study is to evaluate to what extent our strategy is capable of identifying potential failing students in only 30 days. To explain our evaluation design, we first present the participants and material in Section 4.1. Then, we detail the procedure we use during the evaluation in Section 4.2.

4.1. Participants and Material

The participants of our study are students of introductory programming courses at the Federal University of Alagoas, Brazil. We ministered these courses during 3.5 years and collected the metrics we detail in Section 3.2 of each student by using Huxley. The professor encouraged all students to use Huxley to practice by solving the available problems. The use of Huxley was mandatory only during formal exams. Table 1 distribute the number of participants per semester.

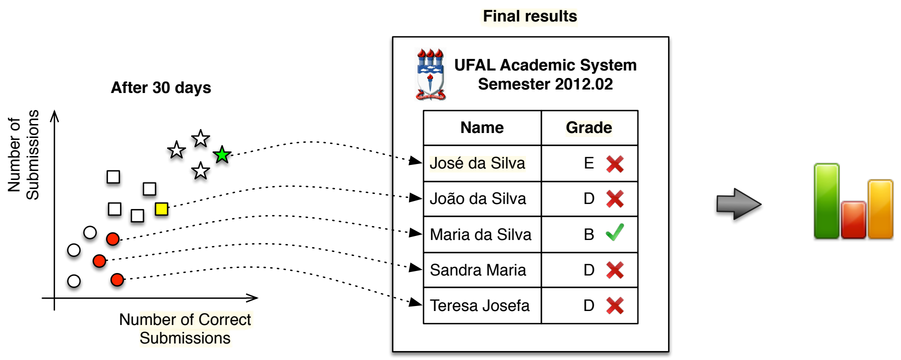
Course	**Number of enrolled students**
2010.02	32
2011.01	38
2011.02	35
2012.01	34
2012.02	29
2013.01	28
2013.02	31
TOTAL:	227

Table 1: Participants per course.

The material of this study consists of almost 300 programming exercises in Huxley. They were available for all students of all courses we use in this paper.

4.2. Procedure

Figure 2 illustrates how we perform our evaluation. The result of executing our strategy consists of groups of students according to the clustering algorithm. Now, according to the groups, we have the potential failing students (see the left-hand side in Figure 2). These students are the ones that have a small number of submissions and correct submissions (represented by circles). Notice we also have students that are potential candidates to successfully pass (represented by stars): due to the high number of submissions to Huxley after 30 days, they seem to be practicing programming really hard. We represent the inconclusive group by squares.



Name	Grade
José da Silva	E ✗
João da Silva	D ✗
Maria da Silva	B ✓
Sandra Maria	D ✗
Teresa Josefa	D ✗

Figure 2: Checking the strategy results against the actual grades.

After applying the strategy, we now need to check whether it correctly predicts the failing students after 30 days. To do so, we use the academic system of the Federal University of Alagoas to look for grades and check whether the students failed or not. For example, for the detached circles, the strategy successfully identified that, after 30 days, two students would not pass and they indeed did not. Notice, however, that we may face false positives. Despite indicating the student *Maria da Silva*[1] as a failing one after 30 days, such a student seemed to improve herself during the semester and she has been approved. In addition, we may also have false negatives. For instance, our strategy does not point the students *José da Silva* and *João da Silva* as failing ones. Although *José da Silva* seems to be one of the best students after 30 days, he failed the course. The several reasons why this happened is out of the scope of this paper.

To better structure and analyze our results and, at the same time, take false positives and false negatives into account, we consider the following two metrics: precision and recall. Precision is the fraction of retrieved students that are relevant, i.e., the students pointed by the strategy that indeed failed the course. We have a perfect precision, 1.0, when every student retrieved by the strategy is relevant (i.e., a failing student), which means we have no false

[1]All names we consider in this paper are fictitious.

positives. Precision focuses on *quality* and *accuracy*. However, the precision says nothing about whether *all* relevant students were indeed retrieved.

$$Precision = \frac{|\{relevant\_students\} \cap \{retrieved\_students\}|}{|\{retrieved\_students\}|}$$

Recall, in its turn, is the fraction of relevant students that are retrieved, i.e., it is the probability of retrieving a failing student. A perfect recall, 1.0, means that we retrieve all failing students, which means we have no false negatives. In this context, recall focuses on *completeness* and *quantity*. Notice that recall says nothing about how many irrelevant students (students that will pass) the strategy retrieved.

$$Recall = \frac{|\{relevant\_students\} \cap \{retrieved\_students\}|}{|\{relevant\_students\}|}$$

To better explain these metrics, consider the detached students in Figure 2 as our set (three circles, one square, and one star). The relevant set is {*José, João, Sandra, Teresa*}, whereas the retrieved set is {*Maria, Sandra, Teresa*}. The intersection set is {*Sandra, Teresa*}. This way, we have

$$Precision = \frac{|\{Sandra, Teresa\}|}{|\{Maria, Sandra, Teresa\}|} = 67\%; Recall = \frac{|\{Sandra, Teresa\}|}{|\{José, João, Sandra, Teresa\}|} = 50\%.$$

Our strategy pointed two out of three students as failed ones and they indeed failed. Therefore, we have 67% of accuracy when identifying potential failing students, raising one false positive. On the other hand, only two out of four students have been pointed as failed ones. This means that the strategy was not able to identify all failing students, raising two false negatives.

Last but not least, after confronting the strategy results with the academic system and summarizing precision and recall, we apply a statistical test to check for significance. Here, we rely on the proportion statistical test based on the Bernoulli distribution [] so that we have a binary distribution: fail or pass. In this paper, we follow the convention of considering a factor as being significant to the response variable when *p-value* < 0.05 [].

5. Results and Discussion

In this section, we describe the results and test our hypotheses before discussing their implications. All data, materials, and R scripts are available at `http://www.ic.ufal.br/`.

5.1. Results

In our evaluation, we use data of 7 courses (3.5 years) totalling 227 students. We apply our strategy by setting k-means to compute two and three groups. We now proceed separately, reporting the results considering both cases.

5.1.1. Two groups

When considering two groups, we set the strategy to consider all students in two categories: fail or pass. Figure 3 illustrates the results for all courses. Notice that Figure 3(b) contains an outlier. In this particular case, the strategy pointed that all students but one would fail the course. Clearly this result is a consequence of such outlier. This way, the presence of one outlier might represent a problem when considering two groups.

To better analyze our results, we remove this outlier and execute our strategy again. By using two groups and properly removing the outlier, we achieve the following results for precision and recall:

$$Precision = \frac{92}{145} = 63\%; \qquad Recall = \frac{92}{115} = 80\%.$$

Here we observe a high recall, i.e., 80%. This means we only miss 20% of the failing students. We have few false negatives, but this result says nothing about how many false positives (irrelevant students) we retrieved. However, notice that this result (80%) represents our particular sample. We now need to find the population proportion, enabling us to generalize our findings. To do so, we need to check for statistical significance by executing the proportion hypothesis test. In this context, the test reveals that the population recall is 73% with a confidence level of 95% (*p-value* = 0.045). Regarding precision, our results show a sample precision of 63%. To generalize, we again execute the test and find that the population precision would be 56% (*p-value* = 0.035).

5.1.2. Three groups

Analogously, we set our strategy to consider three groups. Here, besides the failing and passing groups, there is one extra group where the strategy cannot conclude anything about it. Figure 4 shows the results for three groups. Differently from two groups, here one outlier does not completely compromise the strategy.

We present the precision and recall for three groups in what follows:

$$Precision = \frac{73}{91} = 80\%; \qquad Recall = \frac{73}{115} = 63\%.$$

Now the strategy returns a precision of 80% for the sample proportion. To generalize our results, we again execute the proportion hypothesis test. Regarding the population, our results reveal that the precision is 72% with a confidence level of 95%. In other words, from the set we retrieve, we can identify with statistical significance (*p-value* = 0.04) 72% of the students that indeed will fail the course after 30 days. We repeat this process for recall and find 63% for the sample and 55% for the population, *p-value* = 0.033.

Notice that with three groups the results of precision and recall are inverted when compared to two groups. Our strategy uses k-means, which takes into account two metrics—number of submissions and number of correct submissions—and the number of groups. Both executions are totally independent and the algorithm is not aware of precision, recall, and the academic system. Therefore, we take these results as a coincidence.

5.2. Discussion

In this section we discuss the results. We first discuss the number of groups we should set when using our strategy and then we discuss our achievements regarding final exams.

5.2.1. Two or Three Groups?

When applying our strategy considering two groups, the results indicate a higher recall when compared to three groups. This means that the strategy can identify the majority of the failing students (raising only few false negatives). Although this is an interesting result, the strategy with two groups yields many false positives. In fact, the precision is lower when compared to three groups, i.e., the set we retrieve contains many students that pass. In this situation, professors and assistants might waste effort trying to recover students that actually do not need recovering. On the other hand, with three groups we have a higher precision, which means professors should give special attention to the students the strategy retrieves, since 72% of them tend to fail. Nevertheless, since the recall is lower than with two groups, we have more failing students not retrieved by the strategy for three groups.

In this context, setting the number of groups to execute our strategy seems to depend on the professors priorities and resources. In case the professor has available time and additional assistants to help her, she can probably use two groups, which consists of a more complete retrieved set (the recall is higher for two groups), even though it contains many false positives. However, notice that applying the strategy with two groups is more likely to outliers problems, such the one we depict in Figure 3(b).

On the other hand, if the professor wants to avoid false positives because of no available time or few assistants to help, she might prefer to use three groups, despite being aware of potential failing students not retrieved by the strategy (many false negatives, lower recall than with two groups).

5.2.2. Final Exams

Even though the strategy pointed students as potential failing ones in the first 30 days, some of them passed, raising false positives. Table 2 illustrates the precision for two and three groups as well as the false positives. For two and three groups, the precision is 56% and 72%, respectively. This way, we have 44% and 28% of false positives. As mentioned, the number of false positives is greater when considering two groups.

Groups	**Precision**	**False Positives**	**Final Exam**
2	56%	44% (53 students)	35% (19 out of 53 students)
3	72%	28% (18 students)	33% (06 out of 18 students)

Table 2: False positives (students pointed as potential failing ones but passed) that performed the final exam.

To better understand these false positives, we now analyze their grades at the academic system. In our study, we find that some of these students did not pass in the first place. In fact, they needed to perform the final exam to pass. At the university we focus on this paper, the final exam represents a second chance and is only available to students with not enough grades to pass.

This result is important in the sense that, although the strategy might raise false positives, it seems worth to follow these students as well. We find that at least 33% of them need help (regardless of the number of groups, two or three), otherwise they reach the final exams. This way, professors and assistants can also give special attention to these students, which may improve their learning process, avoid final exams, and lead to better grades.

5.2.3. Back to our objective

As mentioned, the objective of our study is “to evaluate to what extent our strategy is capable of identifying potential failing students in only 30 days.” Given the results we achieve, we believe our strategy is indeed capable of identifying the failing students early. In particular, from the group of students our strategy points as “likely to fail,” 72% of the students indeed fail with 95%

standard confidence level by using three groups.

5.3. Threats to validity

In this section we present the threats to validity of our study. Although our sample is reasonably big (227 students), our study has homogeneities that might pose threats. For example, the same professor for all the 7 courses and the same language used (C language) threats external validity. In this way, it is difficult to extrapolate the results to other contexts. Nevertheless, our strategy uses metrics to identify potential failing students. In this context, we argue that the metrics we use (submissions and correct submissions) do not necessarily depend on factors such as the professors and even less on the adopted languages. However, our claim is not enough and we need further studies to better generalize our conclusions.

The use of Huxley threats internal validity. Students must know how to use the system to submit their solutions. If the students somehow do not get used to the system, they might feel frustrated, hindering their learning and, consequently, biasing our results. We minimize this threat by introducing Huxley in the very first classes as well as by using assistants to help the students on how to use Huxley.

6. Related Work

Reducing the failure rate in programming courses has been achieved in previous work []. The authors performed a study during four semesters. They identified three main factors to reduce the failure rate: (i) using Python as the first introductory programming language, which, according to the authors, may lead the students to better focus on algorithms and problem solving, instead of spending time with advanced concepts of other existing languages at an early stage of the learning process; (ii) using visualization environments to support students and improve the way they can understand abstract concepts related to programming; and (iii) assigning individual problems to each student, hindering students sharing or borrowing solutions to their friends. Like our work, the authors are concerned with the high rate of failure students. However, while they focus on a solution to decrease such numbers, we provide a strategy—evaluated in seven semesters—to early (30 days) identify potential failing students. Then, professors and assistants may act to help these students with additional classes and particular conversations. After studying and understanding why exactly they tend to fail, we then are ready to provide a solution. [achei esse final meio estranho. Nao vendi bem!]

7. Concluding Remarks

This paper presented a strategy able to identify *early* potential failing students in introductory programming courses. We focused on the identification, rather than on approaches to avoid the failings, e.g., changing the programming language. Our strategy consists of three simple steps: the use of an online judge system, the collection of metrics from this system; and the application of a clustering algorithm. In an empirical study using 7 courses (representing 3.5 years), the results suggest that our strategy can early detect (within only 30 days) the failing students, which is a promising result. In particular, from the group of students our strategy points as "likely to fail," 72% of the students indeed fail with 95% standard confidence level. We also found that the remaining set of 28% of students that passed is still interesting: at least 33% of them had difficulties to pass and reached the final exam. This way, some of these students need help as well. In case this happens, they may avoid final exams and achieve better grades at the end of the course.

As future work, we intend to use other metrics and clustering algorithms to better define and analyze our strategy. Additionally, by using the results of our strategy during actual semesters, we intend to make professors and assistants aware of the potential failing students. Then, we are ready to evaluate whether the strategy application can somehow help on reducing the high rate of failing students in introductory programming courses.

8. Acknowledgments

We would like to thank CNPq, a brazilian research funding agency, for partially supporting this work. Márcio's work was partially supported by RHAE 453716/2013-0.

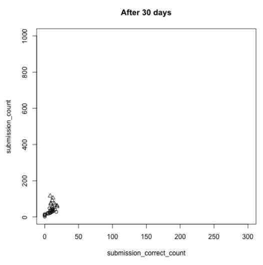
[1] A. Yadin, Reducing the dropout rate in an introductory programming course, ACM Inroads 2 (4) (2011) 71–76. `doi:10.1145/2038876.2038894`.
URL `http://doi.acm.org/10.1145/2038876.2038894`

[2] P. Kinnunen, L. Malmi, Why students drop out cs1 course?, in: Proceedings of the Second International Workshop on Computing Education Research, ICER '06, ACM, New York, NY, USA, 2006, pp. 97–108. `doi:10.1145/1151588.1151604`.
URL `http://doi.acm.org/10.1145/1151588.1151604`

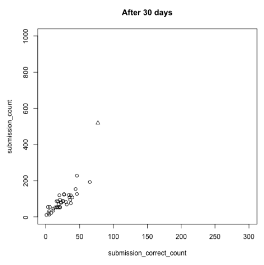
[3] J. Bennedsen, M. E. Caspersen, Failure rates in introductory programming, SIGCSE Bulletin. 39 (2) (2007) 32–36. `doi:10.1145/1272848.1272879`.
URL `http://doi.acm.org/10.1145/1272848.1272879`

[4] T. Jenkins, On the Difficulty of Learning to Program, in: 3rd annual Conference of LTSN-ICS,, 2002.

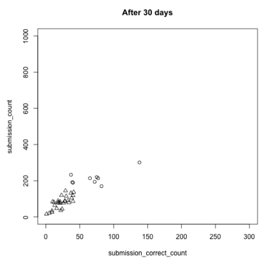
[5] R. d. B. Paes, R. Malaquias, M. Guimarães, H. Almeida, Ferramenta para a avaliação de aprendizado de alunos em programação de computadores, in: Anais dos Workshops do Congresso Brasileiro de Informática na Educação, Vol. 1, 2013, pp. 203–212, in portuguese. `doi:10.5753/CBIE.WCBIE.2013.203`.
URL `http://www.br-ie.org/pub/index.php/wcbie/article/view/2669`



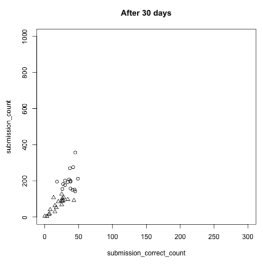
(a) 2010.02



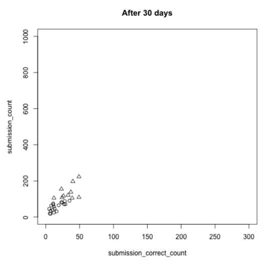
(b) 2011.01



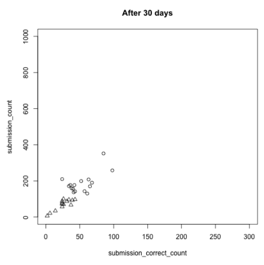
(c) 2011.02



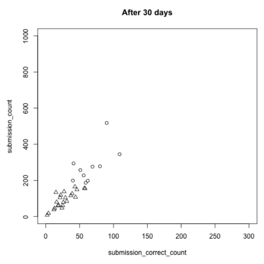
(d) 2012.01



(e) 2012.02

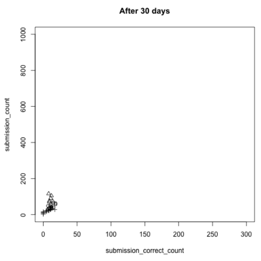


(f) 2013.01

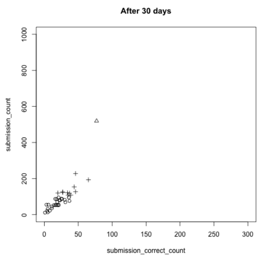


(g) 2013.02

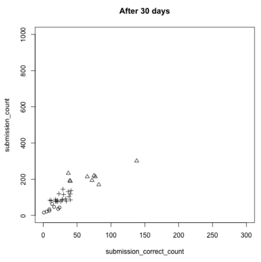
Figure 3: Strategy applied with two groups.



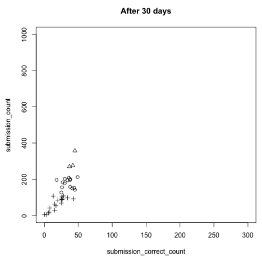
(a) 2010.02



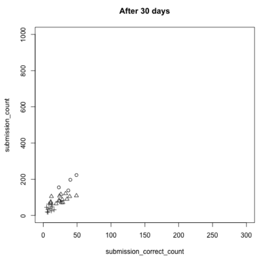
(b) 2011.01



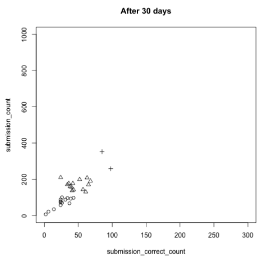
(c) 2011.02



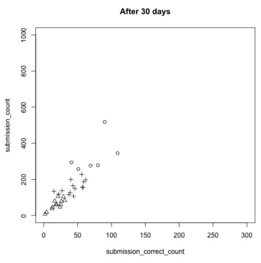
(d) 2012.01



(e) 2012.02



(f) 2013.01



(g) 2013.02

Figure 4: Strategy applied with three groups.