Masaryk University
Faculty of Informatics

Educational Data Analysis for Introductory Programming

Master's Thesis

Matěj Vaněk

Brno, Fall 2018

MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Educational Data Analysis for Introductory Programming

MASTER'S THESIS

Matěj Vaněk

Brno, Fall 2018

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Matěj Vaněk

Advisor: doc. Mgr. Radek pelanek, Ph.D.

Acknowledgements

These are the acknowledgements for my thesis, which can span multiple paragraphs.

Abstract

This is the abstract of my thesis, which can span multiple paragraphs.

Keywords

introductory programming, dashboard

Contents

1 Introduction

E-learning is a modern, developing way how to approach to education using opportunities of computers and the Internet. These conveniences enable teachers and creators of e-learning systems to transmit desired knowledge to much more learners than it would be possible in traditional school lessons. Learners may be from different parts of the world, of different age, education or skills. The ease and relative effectiveness of e-learning systems caused their big development in last two decades – in almost all fields where usage of e-learning seems reasonable, there is some computer educating tool. Programming and computer science in general are not an exception.

While maintaining an e-learning system, every administrator of educational content needs proper information about the performance and characteristics of educational items (tasks) and users (learners). Due to fragmented specialization of e-learning systems domains, variety of possible learning mechanisms and novelty of the e-learning as a subject of science, processes how to obtain this needed knowledge are not standardized.

The aim of this thesis is to find ways how to properly find and express this information about tasks and learners in the field of introductory programming e-learning systems based on data from e-learning system RoboMission. Results of this research is summarized into a form of dashboard which is intended for once-in-a-while use by RoboMission developers and teachers who use this system in their school classes.

Motivation for this research is better understanding to the properties of RoboMission's tasks and learners and thus solid base for improvement of the whole system. Methodological results of this thesis in general are transferable to other introductory programming e-learning systems, where they can be applied after changes with respect to the systems' specific features.

STRUCTURE OF THESIS

2 Research Questions

Based on aforementioned requirements, particular research questions of this thesis were selected as follows:

- What is the difficulty of a task?
- What is the complexity of a task?
- How unique are the correct solutions of a task?
- Is a task similar to some other task?
- What is the learner's performance within a task session?
- What is the learner's performance in the whole system?
- What problems do learners have while solving a task?

SOMETHING ELSE?

3 Theory

SOMETHING

3.1 Introductory Programming E-learning Systems

Introductory programming learning is concerned with the very first moments when a learner meets programming. Its main goal is to make learner familiar with basic principles of programming languages and algorithmic thinking. This includes constructs such as commands, loops, conditions, variables, functions and recursion.

3.1.1 Types of Introductory Programming E-learning Systems

Introductory programming e-learning systems can be divided by two criteria.

The first criterion is the form in which the program is constructed. Two main code-construction ways are textual and block ones.

Textual construction consists in program code writing character by character. Here a learner has to learn both semantic and syntactic aspects of a code structure at once.

Block construction approach is based on program composition from prepared blocks where one block represents one entity such as command or control sequence (often with slots needed for syntactical correctness of given entity). This leads to restricted domain of possible programs and avoidance of syntactic errors, learners learn program syntax only implicitly. Drag-and-drop blocks principle is also more friendly to mobile devices users than typing on a keyboard or display.

The second dividing criterion is the environment in which a program output is processed.

Classical console environment which is used in programming practice, is very minimalist – based on program and input it returns only a textual response. This is very effective but on the other hand, beginners who are used to graphical interfaces may consider this unusual and less user-friendly.

Frequent graphical output environments are two. Drawing interface, in which learner controls a pen (often represented by a turtle

Table 3.1: Examples of Introductory Programming E-learning Systems developed at Faculty of Informatics, Masaryk University

system	language	code construction	output interface	loops	conditions	variables	functions	recursion
RoboMission	Python-like	block	grid	*	*			
Turtle Graphics	custom textual	block	drawing	*		*		
Robotanist	pictorial	block	grid				*	*
Interactive Python	Python	textual	console	*	*	*	*	?

and its tail) by series of distance and direction-change commands, and grid interface, where a robot walks and performs actions on squares of grid game world.

This programming e-learning systems division can be demonstrated on systems developed by Adaptive Learning research lab at Faculty of Informatics, Masaryk University, Brno, Czechia[1], as seen in table 3.1.

3.1.2 Measures

In order to obtain information about measures and characteristics used for displaying description of tasks and learners data if the field of introductory programming e-learning systems, a research of literature was performed. We found 15 articles from last 11 years which are at least partially concerned with this topic. We found 41 data measures and characteristics mentioned in articles, they can be seen in tables 3.2, 3.3 and 3.4. This set comes from e-learning systems of various types of content and logging and therefore only a part of these items is at least theoretically usable in context of RoboMission – the usable ones are marked as "relevant".

1. http://https://www.fi.muni.cz/adaptivelearning/

FOOTNOTES IN TABLES

Based on this survey and given research questions, our own measures and characteristics were chosen. Some of them came from the literature overview, some were adapted with minor modification, the others was created literature-independently in order to cover RoboMission specifics and fill gaps among found measures.

Measures and characteristics related to each research question are described in corresponding chapters of this thesis.

3.2 Dashboards

During a research of so far created introductory programming dashboards in literature, there was found no dashboard of our type, i. e. designed for non-real-time, once-in-a-while use.

REAL-TIME OR OTHER DASHBOARDS: DIANA, MURPHY, MATSUZAWA-OSA, MATSUZAWA-PROGRAMMING

DASHBOARDS IN GENERAL: PODGORELEC, SANTOS, FEW

CA. 1 PAGE

Table 3.2: Found measures and characteristics, part 1/3

no.	name	papers	relevant
TIME			
01	working time	[1], [2], [3], [4]	*
02	time between submissions/compilations	[5]	*
03	time between compilation errors	[5]	
04	percentage of time in non-compiling state	[1], [2]	
05	compile error correction time	[2]	
06	block-editing-mode time ratio	[2]	
KEYSTROKES, CHARACTERS, LINES			
11	total keystrokes	[1]	
12	percentage of keystrokes in non-compiling state	[1]	
13	total lines of code	[1], [2], [6], [7]	*
14	code update pattern[a]	[8]	
15	time change of code update pattern	[8]	
OPERATIONS			
21	particular submissions/compilations	[3]	*
22	total submissions/compilations	[3], [9]	*
23	total successful/unsuccessful compilations	[3]	
24	percentage of successful/unsuccessful compilations	[3]	

Table 3.3: Found measures and characteristics, part 2/3

no.	name	papers	relevant
CODE CONTENT			
31	total control flow statements	[1], [6]	*
32	total function definitions	[6]	
33	total variables	[6]	
ERRORS			
41	particular compilation errors	[3]	
42	most common compilation/run-time errors	[3]	
43	total compilation errors	[3]	
44a	error quotient[a]	[5], [10], [11]	
44b	Watwin (error quotient improvement)[b]	[10], [11]	
45	error category	[4]	*
SCORE			
51	score/points/grade	[5], [6], [8], [10], [12], [13], [14]	*
CORRECTNESS			
61	correctness	[4], [10]	*
62	tests passed	[10]	
UNIQUENESS			
71	unique unit test outputs	[9]	
72	unique AST	[9]	*

Table 3.4: Found measures and characteristics, part 3/3

no.	name	papers	relevant
TASK SIMILARITY			
81	bag of words of task description	[8], [13]	*
82	API calls similarity	[8], [13]	*
83	abstract syntax trees similarity	[8], [13]	*
84	concepts contained similarity	[15]	*
85	levenshtein distance	[15]	*
86	n-grams overlap	[15]	*
87	input-output behavior similarity	[15]	*
88	program dependence graph structure	[15]	
89	code-state transition graph node similarity	[12]	*
MODE OF STATISTICS			
91	absolute stats for learner	all	*
92	relative stats for learner[a]	[3]	*
93	absolute stats for class	[3]	

4 RoboMission and Data

RoboMission [1] is a web e-learning system which has been developed by Tomáš Effenberger et al. at Faculty of Informatics, Masaryk University, Brno, Czechia [16]. It learns its users basic principles of algorithmic thinking such as loops and conditions, in Python-like programming language with visual drag-and-drop interface. A program is created by dragging and dropping prepared blocks from Blockly[2] JavaScript library. Snapshot of RoboMission task solving interface can be seen in the figure 4.1.

The main task of a learner is to navigate a spaceship through a square grid gameboard. In each step, the spaceship moves one row forward; a learner determines the mode of a move (forward, shoot and forward, diagonally to the left, diagonally to the right). Obstacles on the way comprise of asteroids (spaceship may crash), smaller meteoroids (asteroids which can be shot down), wormholes (transport spaceship to other place), diamonds (all of them must be collected) and length-of-program limit.

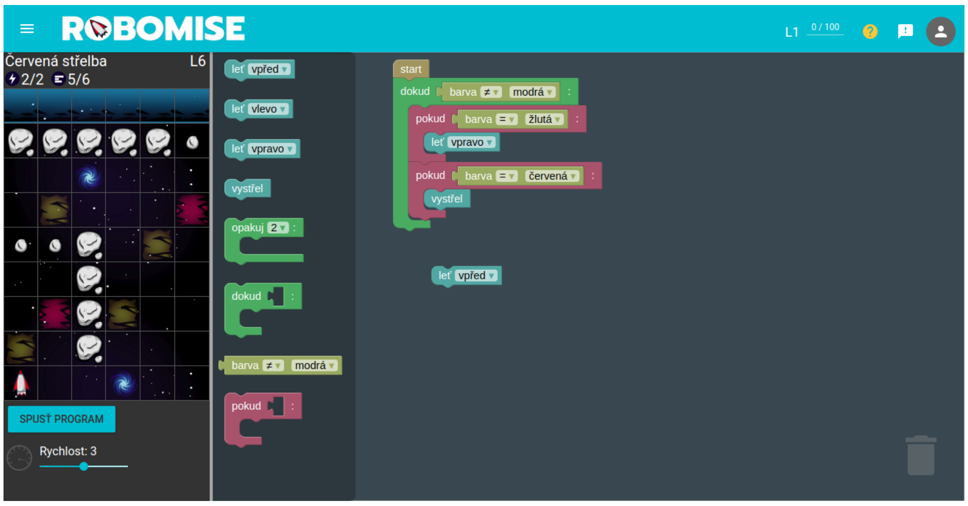
4.1 Data

Provided data consist of three relevant .csv files which described tasks, learner's task sessions and program states logs. The term "task session" refers to one attempt (session) of a learner to solve a task.[3]

Program states are logged whenever learner performs one of two possible actions with code – edit or submission. The code is executed and evaluated only with submissions.

REST OF DATA, CA. 1 PAGE

1. Web site https://www.robomise.cz/ ; currently only in Czech.
2. Web site https://developers.google.com/blockly/ .
3. A learner may have 0, 1 or (rarely) more task sessions of one task.



```
start
dokud barva ≠ modrá :
    pokud barva = žlutá :
        leť vpravo
    pokud barva = červená :
        vystřel

leť vpřed
```

Figure 4.1: RoboMission task snapshot

Table 4.1: Data statistics

Learners	???
Tasks	???
Task sessions	???
Program state logs	???

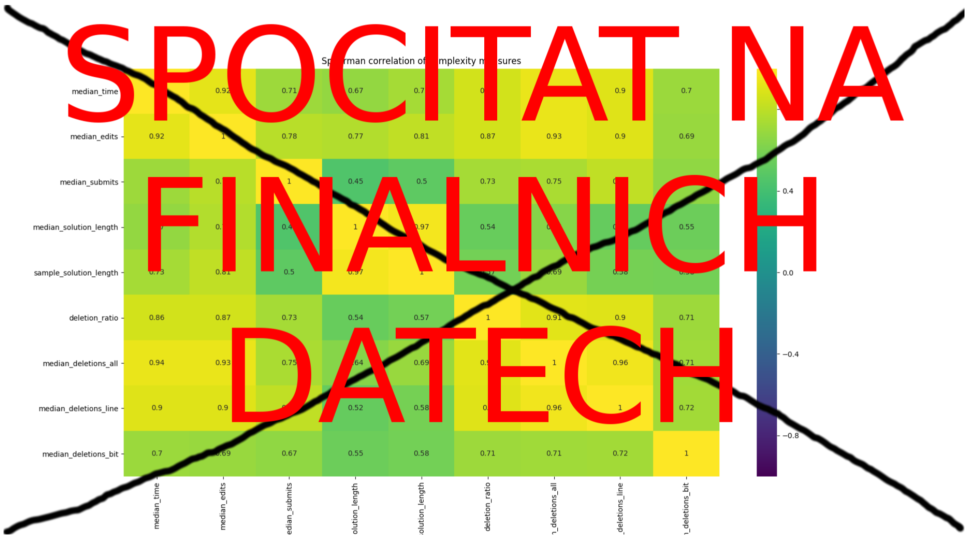


Figure 4.2: Some data statistics graph

5 Task Difficulty and Complexity

Our concept of difference between task difficulty and task complexity follows on from work of J. Sheard et al. [7]. While task difficulty lies in total demands necessary to complete a task, task complexity concerns only with cognitive demands. Difficulty is thus always greater than complexity. Non-cognitive aspects which determine the relationship between difficulty and complexity, are length of task, task description and solving environment.

We selected nine measures which are associated with task difficulty and divided them into two groups by the fact whether they are in most cases influenced only by non-cognitive aspects of difficulty or not.

Task complexity measures which are independent of the other aspects of difficulty:

- ratio of successful task sessions,
- number of distinct blocks used in task's sample solution[1].

Task difficulty measures which are influenced by non-cognitive aspects of difficulty:

- median of solving times,
- median of number of edits,
- median of number of submissions,
- median of length of correct solution,
- length of sample solution,
- median of deletions,
  - number of all blocks deleted,
  - number of all deleting edit actions,
  - deletion used (binary variable),
- ratio of learners who were deleting during the task solving.

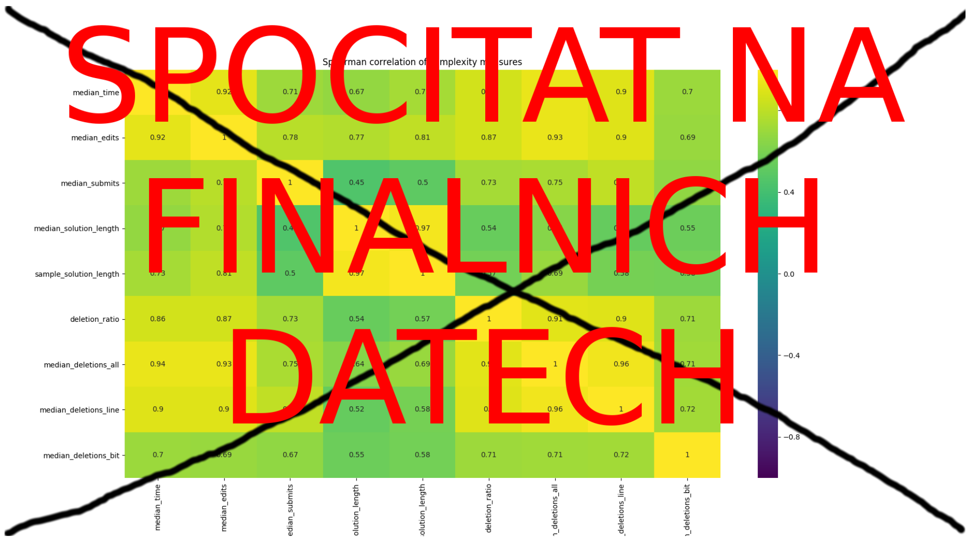


Figure 5.1: Difficulty and complexity measures correlation

THERE IS NO SIGNIFICANT DIFFERENCE BETWEEN DIFFICULTY AND COMPLEXITY IN ROBOMISSION TASKS.

ANALYSES, CA. 2 MORE PAGES

1. Proxy measure to number of necessary concepts.

6 Solution Uniqueness

Solution uniqueness measures:

- number of unique correct solutions,
- entropy of unique correct solutions distribution,
- number of unique visited squares sequences,
- entropy of unique visited squares sequences distribution,
- number of correct solutions clusters,
- sample solution is the most frequent correct solution (binary variable).

ANALYSES, CA. 2 PAGES

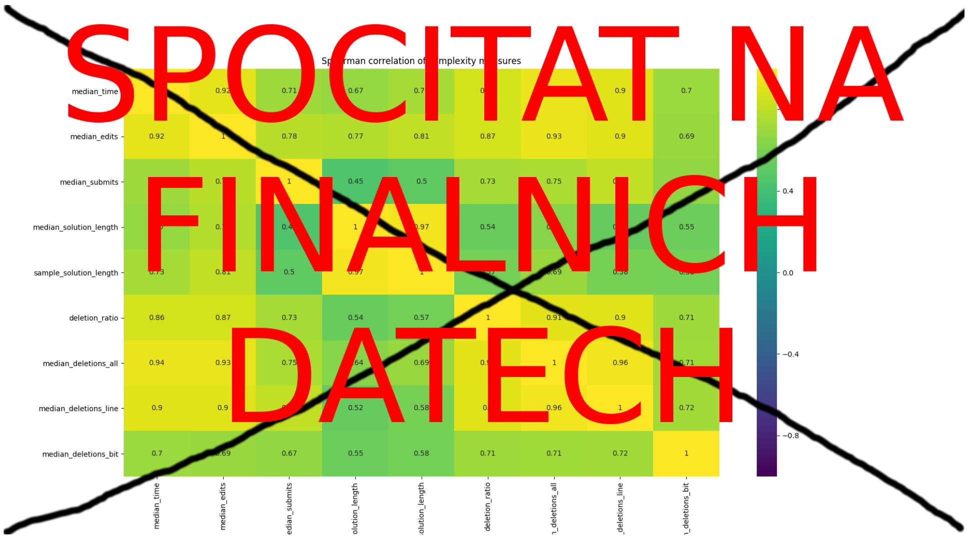


Figure 6.1: Solution uniqueness measures correlation

7 Task Similarity

Task similarity measures:

- sample solution similarity,
    - abstract syntax trees – tree edit distance,
    - bag-of-blocks – euclidean distance,
    - plain program – block Levenshtein distance,
- game world similarity,
    - bag-of-game-world-entities – euclidean distance.

THIS ALL CARTESIAN PRODUCT WITH:

- distance to the closest task,
- how many tasks are situated within 1st, 5th and 10th percentile of distances.

HUANG HAS AST THRESHOLD = 5 TOO
ANALYSES, CA. 2 PAGES

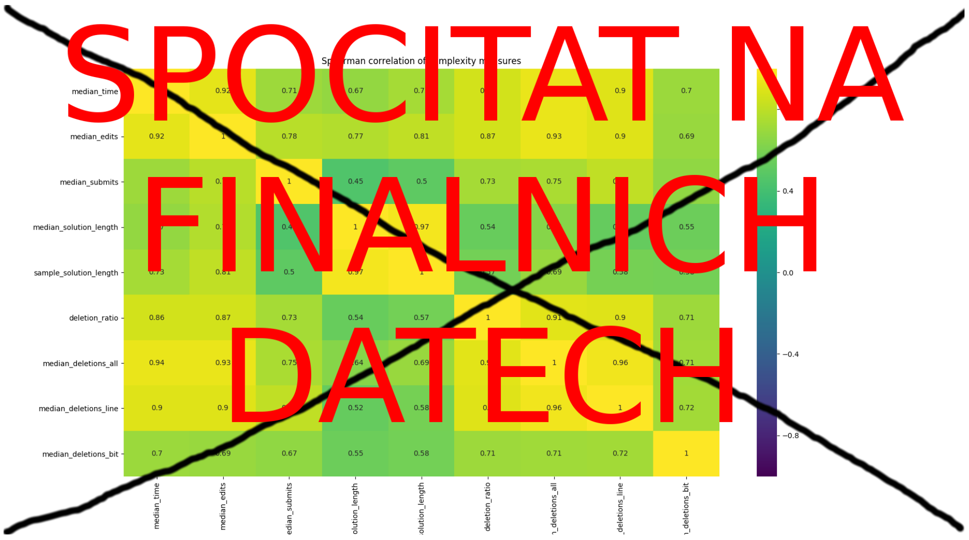


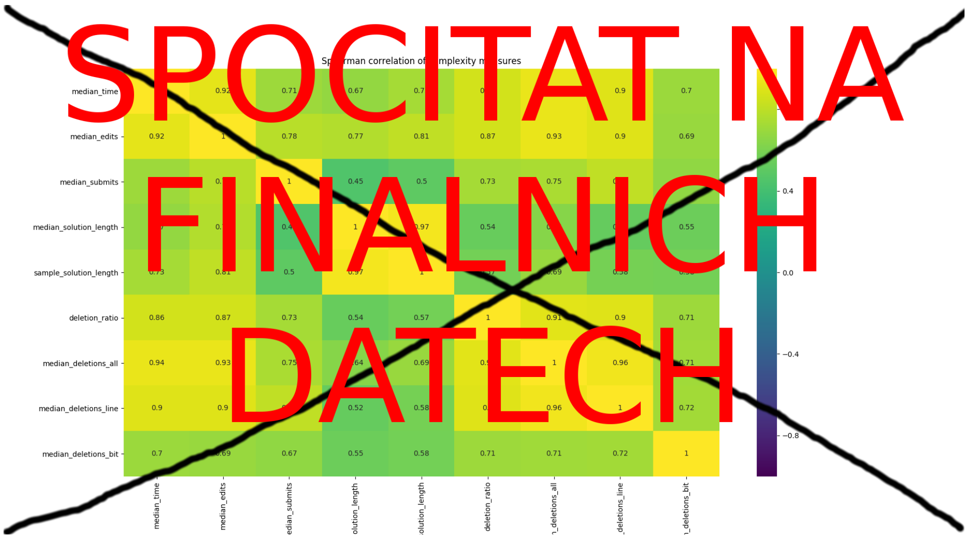
Figure 7.1: Task similarity measures correlation

8 Task Session Performance

User task session performance measures:

- correctness (binary variable),
- solving time,
- number of edits,
- number of submissions,
- deletions,
  - number of all blocks deleted,
  - number of all deleting edit actions,
  - deletion used (binary variable).

ANALYSES, CA. 2 PAGES



SPOCITAT NA FINALNICH DATECH

Figure 8.1: Learner's task performance measures correlation

9 Overall Performance

Overall user performance in the whole system measures:

- total number of solved tasks,
- total number of points gained,
- total number of unique blocks used in correct solutions[1],
- total solving time.

ANALYSES, CA. 2 PAGES

1. Proxy measure to number of completed concepts.

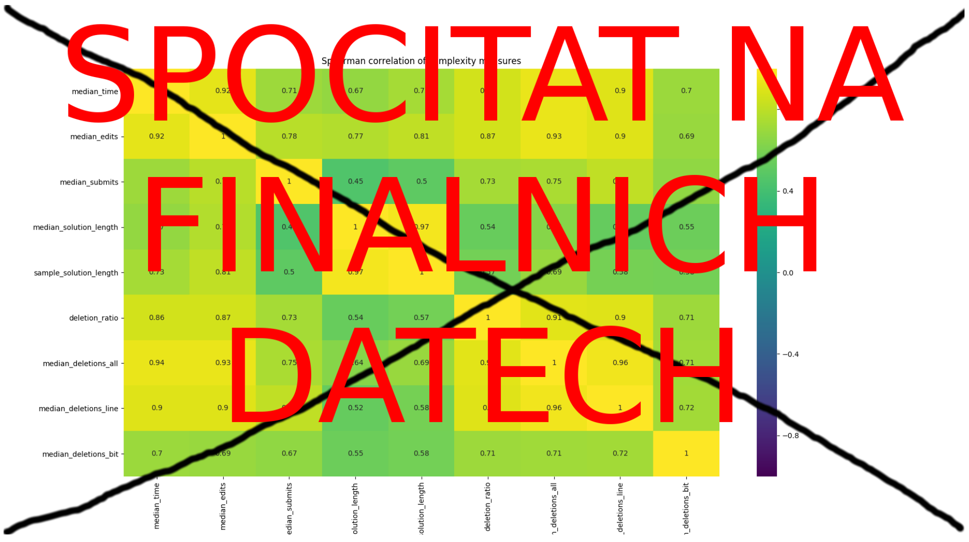


Figure 9.1: Learner's overall performance measures correlation

10 Frequent Problems

Frequent problems characteristics:

- frequent wrong answers,
- frequent stuck points[1].

ANALYSES, CA. 2 PAGES

1. Stuck point refers to the last program code state of unsuccessful task sessions.

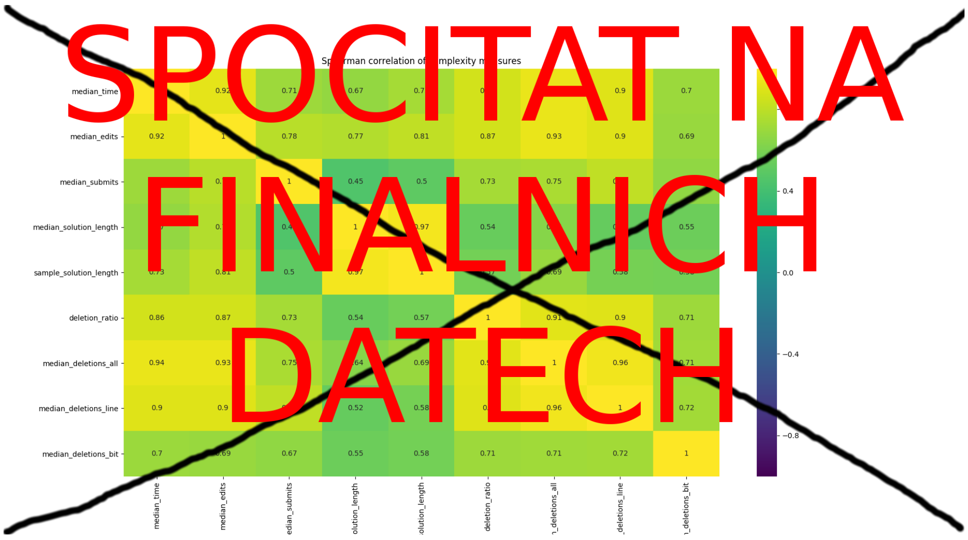


Figure 10.1: Something based on gridplot of all wrong answers

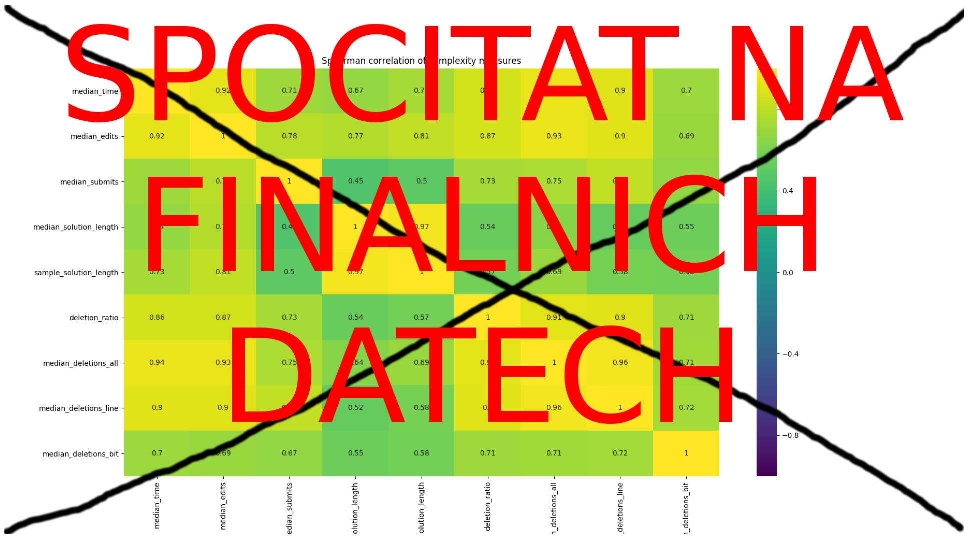


Figure 10.2: Something based on gridplot of stuck points

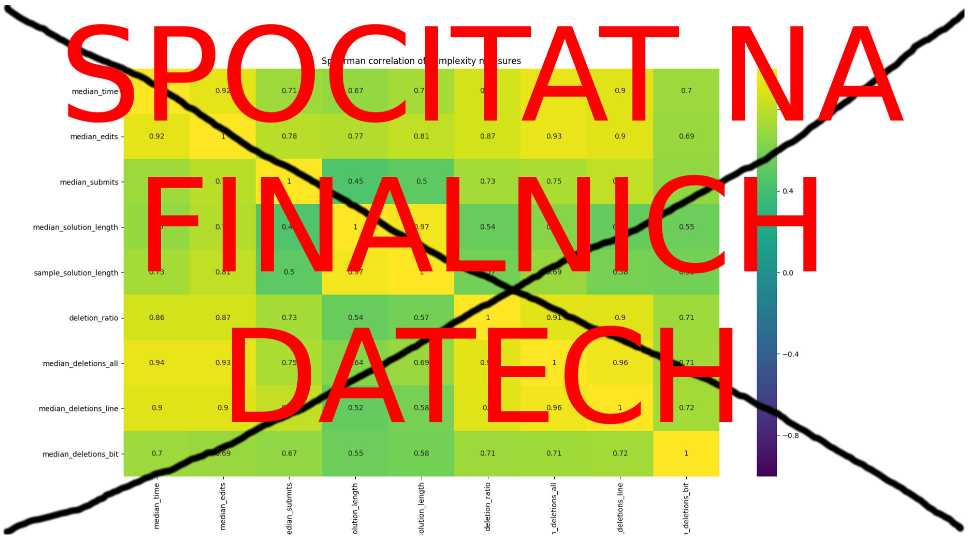


Figure 10.3: TODO: Something about clustering wrong answers

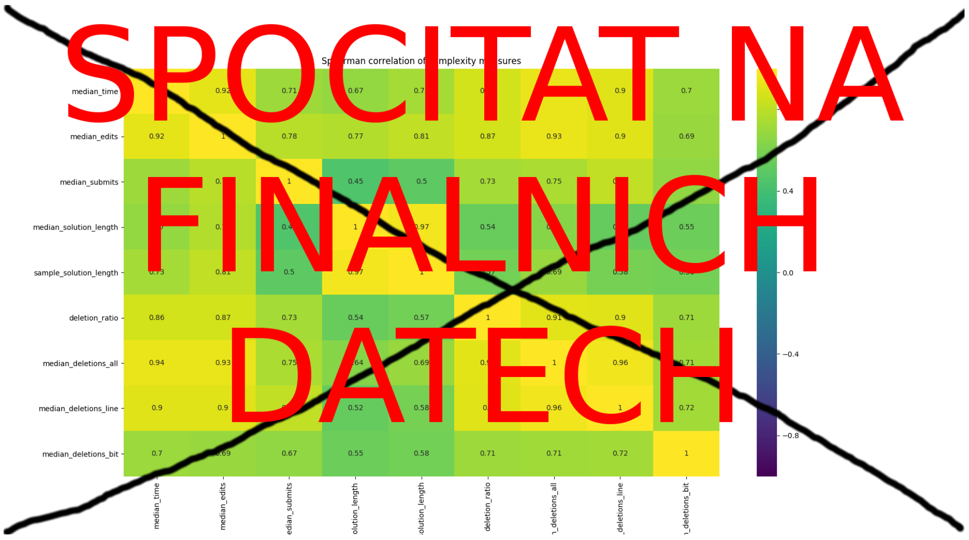


Figure 10.4: TODO: Something about clustering stuck points

11 Dashboard

DESCRIPTION OF DASHBOARD, CA. 2 PAGES

12 Conclusion

WE PERFORMED ANALYSES AND MADE A DASHBOARD, 1-2 PAGES

A Appendix

CODE
DASHBOARD

Bibliography

1. IHANTOLA, Petri; SORVA, Juha; VIHAVAINEN, Arto. Automatically Detectable Indicators of Programming Assignment Difficulty. In: RUHTERFOORD, Becky et al. (ed.). *SIGITE '14 Proceedings of the 15th Annual Conference on Information technology education*. New York: ACM, 2014, pp. 33–38.
2. MATSUZAWA, Yoshiaki; TANAKA, Yoshiki; SAKAI, Sanshiro. Measuring an Impact of Block-Based Language in Introductory Programming. In: BRINDA, Torsten et al. (ed.). *Stakeholders and Information Technology in Education: IFIP Advances in Information and Communication Technology*. Cham: Springer, 2016, pp. 16–25.
3. MURPHY, Christian et al. Retina: Helping Students and Instructors Based on Observed Programming Activities. In: FITZGERALD, Sue et al. (ed.). *SIGCSE '09 The 40th ACM Technical Symposium on Computer Science Education*. New York: ACM, 2009, pp. 178–182.
4. PELÁNEK, Radek. Exploring the Utility of Response Times and Wrong Answers for Adaptive Learning. In: FITZGERALD, Sue et al. (ed.). *L@S '18 Proceedings of the Fifth Annual ACM Conference on Learning at Scale*. New York: ACM, 2018.
5. JADUD, Matthew C. Methods and Tools for Exploring Novice Compilation Behaviour. In: ANDERSON Richard; Fincher, Sally A.; GUZDIAL, Mark (eds.). *ICER '06 Proceedings of the second international workshop on Computing education research*. New York: ACM, 2006, pp. 73–84.
6. ALVAREZ, Andres; SCOTT, Terry A. Using Student Surveys in Determining the Difficulty of Programming Assignments: The Next Decade for Digital Libraries. *Journal of Computing Sciences in Colleges*. December 2010, vol. 26, pp. 157–163.
7. SHEARD, Judy et al. How Difficult are Exams?: A Framework for Assessing the Complexity of Introductory Programming Exams. In: CARBONE, Angela; WHALLEY, Jacqueline (eds.). *ACE '13 Proceedings of the Fifteenth Australasian Computing Education Conference*. Darlinghurst: Australian Computer Society, 2013, pp. 145–154.

8. BLIKSTEIN, Paulo et al. Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming. *Journal of the Learning Sciences*. November 2014, vol. 23, pp. 561–599.

9. HUANG, Jonathan. Syntactic and Functional Variability of a Million Code Submissions in a Machine Learning MOOC. In: WALKER, Erin; LOOI, Chee-Kit (eds.). *Proceedings of the Workshops at the 16th International Conference on Artificial Intelligence in Education AIED 2013*. Aachen: CEUR-WS.org, 2013, pp. 25–32.

10. AHADI, Alireza et al. Exploring Machine Learning Methods to Automatically Identify Students in Need of Assistance. In: DORN, Brian; JUDY, Sheard; QUINTIN, Cutts (eds.). *ICER '15 Proceedings of the eleventh annual International Conference on International Computing Education Research*. New York: ACM, 2015, pp. 121–130.

11. WATSON, Christopher; LI, Frederick W. B.; GODWIN, Jamie L. Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior. In: *ICALT '13 Proceedings of the 2013 IEEE 13th International Conference on Advanced Learning Technologies*. Washington, DC: IEEE, 2013, pp. 319–323.

12. DIANA, Nicholas et al. An Instructor Dashboard for Real-Time Analytics in Interactive Programming Assignments. In: WISE, Alyssa et al. (ed.). *LAK '17 Proceedings of the Seventh International Learning Analytics Knowledge Conference*. New York: ACM, 2017, pp. 272–279.

13. PIECH, Chris et al. Modeling How Students Learn to Program. In: KING, Laurie S. et al. (ed.). *SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education*. New York: ACM, 2012, pp. 153–160.

14. PODGORELEC, Vili; KUHAR, Saša. Taking Advantage of Education Data: Advanced Data Analysis and Reporting in Virtual Learning Environments. *Electronics Electrical Engineering*. October 2011, vol. 114, pp. 111–116.

15. WALENSTEIN, Andrew et al. Similarity in Programs. In: KOSCHKE, Rainer; MERLO, Ettore; WALENSTEIN, Andrew (eds.). *Dagstuhl Seminar Proceedings: Duplication, Redundancy, and Similarity in Software*. Wadern: IBFI, 2007. No. 06301.

16. EFFENBERGER, Tomáš. *Adaptive System for Learning Programming*. 2018. Master's thesis. Masaryk University.