# Evaluating Solution Correctness and Problem Difficulty Based on Code Metrics

**Gurpreet Singh (*# 250674134*)** GSINGH95@UWO.CA
The University Of Western Ontario

## Abstract

**Discovering the effects of code style on code functionality and problem structure.** This study aims to showcase how well formatted, reusable and maintainable code is fundamentally better in all circumstances by looking at over 1 million results from a competitive programming website and analyzing the correlation between question and solution. The goal is to promote code quality checking in online 'just in time' teaching resources to improve the performance of interviewers, researchers, and students.

## 1. Description of Applied Problem

### 1.1. Problem

A large amount of educational material related to programming exists on the internet but the majority of which is not well structured or presented. An applied problem that can be observed from educational material found online is that code quality is often left mutually exclusive from code functionality. Astrachan (2004)

This leads to some students believing it is acceptable to write code that produces the correct result even if the process behind it is not correct. Online code challenge websites like CodeChef.com do not take into account the style and quality metrics of a code submission when judging competitions. CodeChef (2017) Cutting corners in the learning process advances into a complete disregard for best practices in open source software and in the workplace which results in a larger amount of errors. Readability of code is an essential metric in software engineering and can be improved even with simple additions of whitespace between lines. Buse and Weimer (2010)

Computer code written by researchers and other individuals who are just trying to accomplish a result in any way possible is often of the worse quality. This is because they learn using the 'just in time' mentality and the resources online that promote this mentality disregard code quality. Astrachan (2004) Lack of code quality directly reduces it's re-usability because other programmers have a harder time understanding what the code is doing. If these teaching resources could use questions and checks that promote better code quality, many technical innovations could be made. Researchers would be more educated on how to create reusable code and this would influence developers to take their ideas and apply them to real life use cases. Gandhi and Bhatia (2010)

Code quality post processing software is often used in production development environments to ensure good style choices. These checks are much less useful at this senior level than they would be at an educational level. If programming style can be judged on a submission, companies conducting technical interviews will be able to better judge applicants and make a more informed decision.

This study will focus on proving that code quality can have an influence on code functionality, as well as which kinds of questions influence good or bad code styles.

### 1.2. Proposed Solution

A solution to these problems is linking the scoring process in programming problems to a metric derived from running code quality checks on the submission.

Not only will this analysis benefit educational institutes but also companies and competitions that judge people on their code submissions.

Project report for CS4437/CS9637: Intro to Data Science. University of Western Ontario, Winter 2017.

# 2. Description of Available Data

## 2.1. CodeChef Dataset

CodeChef.com is a competitive programming web application that has posted all of their questions and solutions onto the data science website, Kaggle. The data consists of a questions comma separated file, a solutions comma separated file and 3 files that show the code associated with each solution id. The set contains about 1000 problem statements and over 1 million code solutions submitted. This should be a more than sufficient to make a training and test data set.

The important features available for each question are:

- title
- link
- difficulty level
- question statement
- time limit

The important features available for each solution are:

- status (correct or wrong)
- time taken
- memory taken
- language written in
- solution url

## 2.2. Filtering by popular languages

The code submissions are written in many different programming languages and each language has it's own code analysis tool. Therefore, to make the process simpler and come up with higher quality results, the data will need to be filtered by the top languages used. Figure˜1 shows that C++, Java, C and Python are the most popular submissions in this dataset. There are 4 versions of C++ but it should be possible to process them with one tool.

Wang (2016)

# 3. Plan for Analysis and Visualization

## 3.1. Description

In order to make the connection between code functionality and code style, a large part of the experiment is obtaining a accurate and unbiased metric for code quality. Since different tools produce different metrics, a way to normalize the results into a comparable format will need to be discovered. Some options for code quality tools are: CMetrics, cpplint, CScout, Findbugs, ckjm, pydocstring, xenon and pylint. Once a code quality metric is associated with each solution, I will aim to discover the correlation between that metric and the difficulty level stated on CodeChef. From those statistics I could use correlation matrices as visualizations to show how accurate the CodeChef difficulty level is.

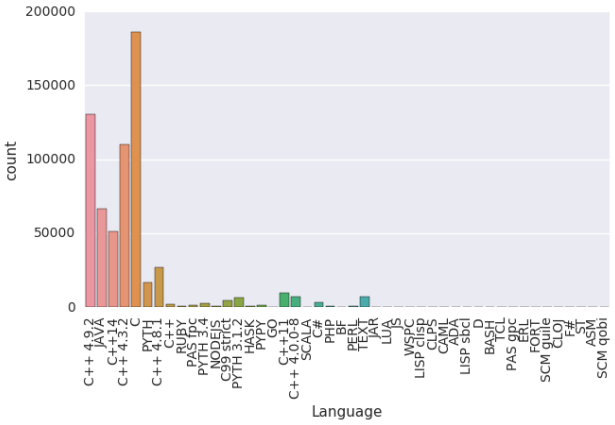


*Figure 1.* Frequency of each programming language that occurs in the dataset of solutions

## 3.2. NLP on Questions

Another possible analysis could be made by processing the question statements using natural language processing and see what kinds of statements are associated with good code style and what kinds of statements result in more failures. Wang (2016)

## 3.3. Code Equivalence

Using some set code metric number that divides the solution set into two, high quality and low quality, we can test to find a common structure within these submissions to pin point where most contestants are running into quality problems. Sharma (2016)

Once a set of common structures are defined, this could also be tied back in to the NLP data we aggregated to discover trends showing what question types produce each structure of code. The end result could be visualized with a classifier, classifying questions into solution structure categories.

## 3.4. Conclusions

From this analysis we should be able to conclude how much of an impact good code style has on achieving high functionality and how different question types

might skew the chances of high quality or low quality code.

## References

Owen L Astrachan. Non-competitive programming contest problems as the basis for just-in-time teaching. In *Frontiers in Education, 2004. FIE 2004. 34th Annual*, pages T3H–20. IEEE, 2004.

Raymond PL Buse and Westley R Weimer. Learning a metric for code readability. *IEEE Transactions on Software Engineering*, 36(4):546–558, 2010.

CodeChef. Codechef competitive programming, October 2017. URL http://www.codechef.com. [Accessed Oct 26, 2017].

Parul Gandhi and Pradeep Kumar Bhatia. Reusability metrics for object-oriented system: An alternative approach. *International Journal of Software Engineering (IJSE)*, 1(4):63–72, 2010.

Arjoon Sharma. Program equivalence for problems, October 2016. URL https://www.kaggle.com/arjoonn/program-equivalence-for-problems. [Accessed Oct 26, 2017].

Justin Wang. Nlp and ml experiments, December 2016. URL https://www.kaggle.com/justwjr/nlp-and-ml-experiments/notebook. [Accessed Oct 26, 2017].