For our final project, we performed a statistical analysis of several different features on the popular programming competition website TopCoder. We gathered a large amount of data from the website using automated web scraping and analyzed a wide variety of measures, including comparisons of programming language usage, programming language success rates, challenge rates, programming times, statistics related to programmers’ countries of origin, correlations between successful coding and challenging, and site usage over time.

Overview

TopCoder is a website where users compete against each other in programming “matches.” Competitors are separated into two divisions based on past performance, Div1 and Div2. Users with a higher rating are placed in Div1. At the start of a typical match, each competitor is presented with the same set of three problem statements at three tiers of difficulty with correspondingly different point values, each of which gives a specification of a program they must write within an allotted time limit. After all competitors have submitted their code, they are given a chance to view others’ solutions and “challenge” them if they think the solution is incorrect. Users’ final scores are then calculated based on the problems successfully solved, the time it took them to solve each problem, and their challenges of others’ code.

A problem statement typically includes some background story to give the problem context, a precise (often mathematical) description of the behavior the program must have, a domain of possible inputs, the expected format for program output, and several example input and output pairs (often with a short explanation for why the output is what it is). Each user can select their own programming language preference from a list of several languages, and receive a problem statement tailored to that language with input and output types given in data structures common for that language (for example, vector<string> for C++ versus String[] for Java). Programs are also given time and memory constraints, typically 2.000 seconds of execution time and 64 megabytes of memory. The difficulty ranges from relatively trivial for the lowest-tier problems (such as writing a Huffman decoder given an input string and encoding), to the highest-tier problems which often require an obscure mathematical insight in order to write a program that can operate within the available time and memory constraints.

After all users have submitted solutions, they are also given a chance to view all the other solutions and challenge any which they believe to be incorrect. Upon challenging a program, that program is tested against an extensive list of predefined input and outputs. Any output which differs from the expected output causes the program to be considered incorrect. If a challenge is successful (meaning at least one output differed from the expected output), the challenger is awarded 50 points and the person who submitted the incorrect program receives no points at all for their submission. Otherwise, the challenger loses 25 points.

Finally, any program which has not already been successfully challenged is tested automatically. Again, any program which differs in at least one output is disqualified and receives no points. Each user’s final score is calculated as the sum of their scores for each problem, plus any points or deductions for successful or unsuccessful challenges. Points for each problem are calculated as the problems base score, minus a factor dependent on the amount of time between the moment the user opens the problem statement to the moment their solution is submitted.