Relative Feedback Increases Disparities in Effort and Performance in Crowdsourcing Contests: Evidence from a Quasi-Experiment on Topcoder

Milena Tsvetkova[a], Sebastian Müller[a], Oana Vuculescu[b], Haylee Ham[c,d], and Rinat Sergeev[c,e]

[a] Department of Methodology, London School of Economics and Political Science, London WC2A 2AE, United Kingdom

[b] Department of Management, Aarhus University, 8210 Aarhus, Denmark

[c] Laboratory for Innovation Science at Harvard, Harvard Business School, Boston, MA 02134

[d] Institute for Quantitative Social Sciences, Harvard University, Cambridge, MA 02138

[e] Harvard Business School, Boston, MA 02163

Abstract

Rankings and leaderboards are often used in crowdsourcing contests and online communities to motivate individual contributions but feedback based on social comparison can also have negative effects. Here, we study the unequal effects of such feedback on individual effort and performance for individuals of different ability. We hypothesize that the effects of social comparison differ for top performers and bottom performers in a way that the inequality between the two increases. We use a quasi-experimental design to test our predictions with data from Topcoder, a large online crowdsourcing platform that publishes computer programming contests. We find that in contests where the submitted code is evaluated against others' submissions, rather than using an absolute scale, top performers increase their effort while bottom performers decrease it. As a result, relative scoring leads to better outcomes for those at the top but lower engagement for bottom performers. Our findings expose an important but overlooked drawback from using gamified competitions, rankings, and relative evaluations, with potential implications for crowdsourcing markets, online learning environments, online communities, and organizations in general.

Introduction

Social comparison is deeply rooted in our nature. According to the eminent social psychologist Leon Festinger, we have a powerful drive to gain accurate self-evaluations and we often achieve this by comparing our opinions and skills to those of others (Festinger, 1954). Even more importantly, we tend to seek a better standing in relation to others, that is, we pursue distinction and status (Fiske, 2011; Payne, 2017). Thus, social comparison induces competition (Festinger, 1954; Garcia, Tor, & Schiff, 2013), and competition increases individual effort. Specifically, contests and tournaments, where financial rewards depend on rank performance, elicit more individual effort than piece-wise remuneration schemes and, in fact, drive individual effort expenditure beyond what is rational (Cason, Masters, & Sheremeta, 2010; Connelly, Tihanyi, Crook, & Gangloff, 2014; Dechenaux, Kovenock, & Sheremeta, 2015; Dickinson & Isaac, 1998). As a result, contests can be successfully employed to spur innovation (Wright, 1983). For instance, there is a growing practice of launching online crowdsourced prize-based contests and idea competitions to solve hard scientific problems, optimize company algorithms, or redesign brand identity (Howe, 2008; Lakhani et al., 2013; Travis, 2008). Still, social comparison can induce competitive behavior and increase effort even without financial incentives (Mago, Samak, & Sheremeta, 2016; Messick & McClintock, 1968; Sheremeta, 2010). This is the idea behind using game design elements such as likes, badges, skill levels, and leaderboards to increase learning in online courses, exercise on e-health platforms, or performance in standard organizational tasks (Denny, McDonald, Empson, Kelly, & Petersen, 2018; Koivisto & Hamari, 2019; Landers, Bauer, & Callan, 2017).

Although competition driven by social comparison increases individual effort, it could have detrimental effects on group morale and cooperation. Within-group competition for scarce resources reduces trust and trustworthiness (Charness, Masclet, & Villeval, 2013; Chowdhury & Gürtler, 2015; Liu, Lin, & Xin, 2014), decreases contributions to public goods (Barker & Barclay, 2016), diminishes cooperation (West et al., 2006), and hinders agreement in bargaining situations (Barclay & Stoller, 2014). Competition also increases spiteful behavior and sabotage, making individuals more likely to harm others at a personal cost in order to diminish their opponents' performance and increase their own chances of winning (Harbring & Irlenbusch, 2011).

This study investigates the effects of social comparison on another potential group-level problem—inequality in effort and performance. In creative tasks, input, or effort, is positively correlated with but still distinct from output, or performance. Effort is related to the level of engagement and mental investment, while performance indicates the level of learning, innovation, and creativity that has occurred. Some social

settings benefit from a single best performing contributor or a higher average performance: for example, multiple record-breaking performances can fuel audience interest in a sports tournament, while a single brilliant idea can generate value in an innovation contest. Other social contexts, however, benefit more from an even distribution of effort and performance. Online citizen science initiatives, learning platforms, user-contribution communities, and crowd workplaces flourish when their members keep engaging and learning (Kittur et al., 2013). More generally, reducing experiential inequality has been shown to increase individual productivity and job satisfaction and improve organizational performance (Bapuji, Ertug, & Shaw, 2020; Stainback, Tomaskovic-Devey, & Skaggs, 2010).

We investigate how social comparison affects inequality in effort and performance using data from the online crowdsourcing platform Topcoder. Established in 2001, Topcoder is the largest competitive software development portal that constitutes both a marketplace for software solutions and a social network community. Topcoder maintains an active community of hundreds of thousands of designers, developers, data scientists, and programmers who regularly compete in programming challenges (Archak, 2010; K. J. Boudreau, Lacetera, & Lakhani, 2011; K. Boudreau, Helfat, Lakhani, & Menietti, 2012; Lakhani et al., 2013). Each Topcoder challenge poses a well-defined problem and has a unique leaderboard and prize structure. Some challenges are commercial and offer monetary prizes while others are organized by the platform and only offer contestants the opportunity to gain experience and improve their rating.

Our study focuses on Topcoder challenges where coders iteratively work on their solution and can repeatedly submit it for feedback over the duration of the challenge. To investigate the effects of social comparison, we differentiate between challenges in which the feedback shows performance relatively to the performance of other coders' solutions and challenges in which the feedback is a measure of the performance of the solution in a test. We rely on impressively large data and advanced computational techniques to measure coder ability, effort, and performance and we apply an innovative matching method to offer statistical evidence for causal effects. The results bring attention to an overlooked negative side-effect from exploiting the power of social comparison and relative-performance feedback to motivate user contributions. Our findings suggest the need to carefully evaluate the use of gamified competitions, rankings, and relative evaluations in online communities and organizations that prioritize universal participation and equalized individual outcomes.

Our findings contribute to a growing body of research on contest design for crowdsourced innovation coming largely from economics, management science, and human-computer interaction (Adamczyk, Bullinger, & Möslein, 2012). Researchers have investigated both theoretically and empirically design

elements such as award structures, problem characteristics, contest duration, and evaluation and feedback mechanisms to maximize contestants' effort, predict individual success, or simply understand contestants' behavior (Archak & Sundararajan, 2009; DiPalantino & Vojnovic, 2009; Bullinger, Neyer, Rass, & Moeslein, 2010; LaToza, Chen, Jiang, Zhao, & Hoek, 2015). Several studies have specifically focused on how information about other contestants and their submissions could affect entry, effort, and innovativeness and even more notably, could exert differential effects on different individuals (Archak, 2010; Bullinger et al., 2010; Bockstedt, Druehl, & Mishra, 2016; K. Boudreau et al., 2012; K. J. Boudreau, Lakhani, & Menietti, 2016). What unites these latter studies is the recognition, even if not always made explicitly, that contestant behavior and success are endogenous and shaped by both contest and contestant characteristics. We build upon and extend this research in two important ways. First, by investigating the effects of absolute versus relative feedback, we directly compare situations with weaker versus stronger group effects. And second, by theoretically focusing on inequality, we bring attention to the complete feedback loop, whereby not only the group affects individual behavior, but also individual behavior affects group outcomes.

Theoretical Background and Predictions

We study the effects of social comparison on inequality in effort and performance in online communities, educational platforms, and organizations, which constitute large heterogeneous groups of people. When considering inequality in heterogeneous populations, there are two distinct aspects to take into account—the dispersion of outcomes and the predictability of outcomes (Lynn, Podolny, & Tao, 2009; Salganik, Dodds, & Watts, 2006). Dispersion, which measures the differences between the worst and best performers, represents an intuitive understanding of inequality, as evidenced by the widespread use of the Gini coefficient to discuss income and wealth inequality (Allison, 1978; Piketty & Goldhammer, 2014). In our context, lower inequality would entail more consistent levels of effort and engagement across individuals, resulting in more evenly distributed performance. Predictability, however, is a less straightforward concept because it can indicate both high and low inequality depending on the characteristic that is associated with the outcome. In the case of an inherent quality or virtue, more predictable outcomes imply more equality and fairness (Cook & Hegtvedt, 1983; Jasso, 1983; Salganik et al., 2006). However, in the case of arbitrary or unfair pre-existing differences, more predictable outcomes signal the lack of equality of opportunity and upward mobility (Van de Werfhorst & Mijs, 2010). In a supportive, learning-based community, lower inequality would entail an even chance for everyone to perform well, making outcomes less correlated with prior skills

and achievements. In short, in the context we investigate, lower inequality involves lower dispersion of effort and performance and lower predictability of current from prior performance, while higher inequality entails higher dispersion and predictability.

Previous literature has not directly studied the effects of social comparison on inequality in effort and performance but it nevertheless contains some insights and suggestive empirical evidence, albeit contradictory. On the one hand, in his seminal essay, Festinger suggests that social comparison drives competition in a direction that facilitates convergence towards uniformity (Festinger, 1954). He argues that social comparison increases underachieving individuals' motivation to perform better and hence, they increase effort. In addition, overachievers become willing to help underachievers while, at the same time, underachievers cooperate less with overachievers. Further, he mentions that when the achievement gap is too large, individuals will avoid comparison by simply avoiding or leaving the group. All of these mechanisms—increasing effort, cooperating with underachievers, undermining overachievers, or self-selecting out of an unfavorable comparison group—serve to equalize outcomes and thus decrease inequality in performance. Two of these mechanisms are empirically supported: the tendency to select into and out of contests (Dohmen & Falk, 2011; Eriksson, Teyssier, & Villeval, 2009) and the tendency to sabotage winners in contests (Chen, 2003).

However, empirical research on contests and tournaments contradicts Festinger's expectation that underachievers increase effort. Compared to piece-rate schemes, contests, all-pay auctions, and tournaments result in higher variance in individual effort (Cason et al., 2010; Dechenaux et al., 2015; van Dijk, Sonnemans, & van Winden, 2001). Similarly, the contributions to a public good increase in dispersion when individuals are incentivized with tournament rewards for contributing (Dickinson & Isaac, 1998). These outcomes can be explained with demographic differences and heterogeneous preferences towards winning, risk, losses, and inequality (Dechenaux et al., 2015). In essence, social comparison may have differing effects on different people: some may become more competitive, while others may lose interest and decrease effort as their chances of winning drop. This implies that social comparison could result in a rich-get-richer and poor-get-poorer dynamics that widen existing ability differences and make performance more predictable, going directly against Festinger's arguments. We hypothesize that social comparison tends to do exactly this.

There are two forms of social comparison people can make: upward and downward (Fiske, 2011). Upward social comparison occurs when people compare themselves to those who are better off and although it often triggers negative self-evaluation, resentment, and envy, it can also be motivating and self-enhancing (Collins, 1996). Downward social comparison occurs when people compare themselves to those who are

worse off and, in parallel, it generally boosts self-confidence and improves self-evaluation but may also induce negative affect, especially among those with low self-confidence (Buunk, Collins, Taylor, VanYperen, & Dakof, 1990). We suggest that the positive and negative effects from upward and downward social comparison could align in a way that social comparison becomes motivating and enabling for those who are already well off but demotivating and discouraging for those who are currently worse off.

When individual achievement is measured and evaluated in reference to others, success becomes a moving target. Individuals who put the effort and improve their personal performance may lack a sense of accomplishment because others may still be advancing faster. This may trigger feelings of unfairness, resentment, even relative deprivation among those at the bottom of the chart. These subjective experiences have been linked to short-sightedness, emphasis on small and immediate rewards compared to large and delayed ones (Callan, Shead, & Olson, 2011), and more risky behavior (Payne, 2017). Conversely, those at the top of the chart may experience the opposite effect, whereby a provisional reward for a serendipitous move could increase their confidence and motivation to persevere (Restivo & van de Rijt, 2012). Indeed, prior empirical research shows that higher-ranked individuals behave more competitively than lower-ranked individuals (Garcia, Tor, & Gonzalez, 2006). In short, social comparison exaggerates small differences in ability and effort but individuals' reaction to the ongoing results could feedback into individuals' behavior in a way that reinforces these differences.

We thus expect that social comparison will result in higher individual effort among already well-performing individuals and lower individual effort among worse-performing individuals. This in turn will produce higher inequality by increasing both the dispersion and the predictability of individual performance. In particular, while fierce competition at the top could produce outstanding results and surprise winners, low effort at the bottom is likely to result in lower-quality results from the expected losers. More formally, our predictions are as follows:

H1. Social comparison increases the dispersion of effort by eliciting more effort from top performers but less effort from bottom performers.

H2. Social comparison increases the dispersion of performance by eliciting better performance from top performers but worse performance from bottom performers.

H3. Social comparison increases the predictability of performance from prior performance for bottom performers.

We test these predictions in the context of online crowdsourcing contests using data from programming

challenges on Topcoder.

Data and Methods

Data

We combine data from two sources. First, we obtained detailed data from Topcoder on 189,659 successful code submissions by 17,000 individuals in 367 challenges in the period from 15 December, 2005 until 1 April, 2018. The dataset covers almost all "marathon matches" in the Data Science track that were posted since the platform was launched, except for a handful of challenges with proprietary data. Marathon matches are relatively long-running challenges that allow coders to make repeated submissions and receive provisional scores for their current performance, giving them the opportunity to improve their solutions.

We combine these data with manual classifications of the challenges computed from answers provided by coders in a dedicated challenge on Topcoder itself. Coders classified the challenges along categories such as ease of entry, complexity of the scoring function, type of task, and scoring type. Coders marked the scoring type as **absolute** when "The score shows absolute performance of the solution on the test" and as **relative** when "The score shows performance relative to the performance of the other solutions". We assume that social comparison is more prominent in the relatively scored challenges compared to the absolutely scored ones. To test the hypotheses, we examine how the scoring type of the challenge affects the coder's effort and performance, depending on the coder's prior performance. The data and scripts for the analyses are openly available on Figshare (Tsvetkova, Müller, Vuculescu, Ham, & Sergeev, 2022).

Measures

Our primary measure of effort is lines of code (LOC)—a common metric for programmer productivity and effort employed in software development management (Fenton & Neil, 2000). We calculate the number of code lines (disregarding comments) written by the contestant for the challenge by summing the number of code lines in the first submission with the number of new or edited code lines in all subsequent submissions, given by the code file diffs (see Appendix B). This measure does not differentiate between small edits, such as changing a variable name, and a major overhaul, such as rewriting the algorithm, but it generally captures the amount of work and attention involved. There are two cases where more than 10,000 new code lines were inserted between subsequent submissions, which on closer inspection turned out to constitute hard-coded data; we remove these two outliers from the relevant analyses. To confirm the robustness of

the findings on effort, we additionally look at the number of submissions and the duration of the coder's participation in the challenge, expressed as a proportion of the challenge duration.

To measure the quality of performance, we require an objective metric that does not depend on the scoring type. This precludes using the absolute final scores since they are deliberately scaled for relatively scored challenges, as well as the final rank percentiles since they are influenced by differences in skill-based selection into differently scored challenges. Hence, to quantify the quality of performance, we estimate the cyclomatic complexity of the code in the final submission. Cyclomatic complexity is a software complexity measure that is based on the decision structure of the code (McCabe, 1976). It quantifies the number of independent paths in the source code by counting the number of decision points, which occur at conditionals, logical expressions, and loops. We use cyclomatic complexity to measure how advanced and sophisticated the coder's solution is in terms of control logic (Munson & Khoshgoftaar, 1989). Although the measure increases with modularization and design effort (Shepperd, 1988), very convoluted and inefficient code would also have high cyclomatic complexity. We find that the correlation between the cyclomatic complexity of the code in the final submission and the coder's final rank percentile is positive in the analyzed challenges, with mean Spearman rank correlation 0.51 for absolutely scored challenges and 0.54 for relatively scored challenges, both significantly different from 0 according to one-sample t-tests ($p < 0.000$) but not from each other (two-independent-sample $t = -0.668$, $p = 0.507$). This confirms that the metric is independent from the effects of the scoring type, as intended. Nevertheless, the correlation with rank percentile can be quite low in some cases (Fig. 3, right), which brings attention to the fact that the measure mainly captures the elaboration and development of the code, rather than how well it addresses the particular problem. Relevantly to this, we find that the cyclomatic complexity typically increases with subsequent submissions, further supporting the idea that it reflects elaboration and development, rather than consolidation and refinement (Fig. 3, left). Most importantly, the cross-over experimental design we employ compares the difference in performance between the treatment and control *within* individuals. Essentially, we ignore individual differences in coding style and only focus on the change in the code complexity for each individual.

For the predictability of performance, we estimate the absolute difference between the percentile rank by final score and the percentile rank by prior performance; we then subtract it from 1 in order to have higher values correspond to higher predictability. For prior performance, which we use as proxy for ability, we use the coder's official Topcoder Marathon Match rating. Topcoder calculates this rating using a complex algorithm that evaluates the coder's actual performance in reference to the performance expected

by their previous rating and accounts for how many previous challenges the coder has participated in and how volatile their performance in these previous challenges has been. It should be noted that this measure is not perfect since it is noisier for coders who have participated in fewer previous challenges on the platform. We mitigate for this limitation, however, by controlling for the number of previous challenges in the statistical models.

Coarsened Exact Matching

Since our data are observational, we use a quasi-experimental research design to establish a causal link between scoring type and dispersion of effort, dispersion of performance, and predictability of performance. First, we use coarsened exact matching to identify strata of challenges that are similar in all possible respects except for the scoring procedure. With coarsened exact matching, we first coarsen the matching variables C to fewer meaningful categories $c(C)$ and then perform exact matching on $c(C)$ (Iacus, King, & Porro, 2012). Subsequently, the data is sorted into strata of unique values in $c(C)$. Any stratum that does not include at least one observation for each level of the treatment variable, i.e. scoring procedure, is then pruned. For any estimation that is carried out after the pruning process, we add weights that are based on the relative frequency of observations within strata. This step ensures that differences in the distribution of observations across strata do not distort the results (Iacus et al., 2012).

We choose to match on all variables that might be correlated with the scoring type X and any of the outcomes Y: variables related to the size, duration, difficulty, reward structure, and task of the challenge. For most discrete data, we simply use the levels as the values for which the data must match. For continuous variables we select cut points to coarsen the data based on substantive judgement and an analysis of the variables' distributions. Table 3 in Appendix C summarizes the variables and the coarsened levels that we use for constructing the strata in the matching process.

To coarsen the variables for number of contestants, challenge duration, prize pool, and number of prizes, we started with an *a priori* expectation about meaningfully different categories (e.g., contests without monetary prizes vs. contests with top-three prizes vs. contests with multiple prizes) and adjusted it according to the observed distribution in our sample of challenges (for the same example, we settled on 0 prizes, 1–5 prizes, and 8–10 prizes). For engagement barrier and scoring complexity, which are based on five-level Likert scale responses, we split between the categories "1–Strongly Agree" and "2–Agree" on the one hand, and the rest on the other. The decision how to coarsen the variables for contestants/registrants ratio and average contestant experience was mainly driven by the distributions of these variables. Specifically,

we divided average contestant experience by tercile and split contestants/registrants ratio halfway through, with an additional category for very high values ≥ 0.9.

After performing the matching procedure, we obtain a dataset of 52 challenges in 16 strata, with a reasonable covariate balance for the six coarsened variables (Fig. 4). We note that none of the matched challenges have monetary or nonmonetary prizes (Table 4). This means that they are challenges designed and posted by Topcoder with the purpose to keep their user community stimulated and engaged. Coders contribute to such challenges in order to learn and practice and in order to establish their reputation in the community (K. J. Boudreau et al., 2011; K. Boudreau et al., 2012).

Crossover Quasi-Experiment

Analyzing all the individuals in the matched challenges would not allow us to distinguish between self-selection into the treatment and any actual treatment effect on behavior. To provide causal answers to the hypotheses, we restrict the sample of contest participants to coders with observations for both scoring procedures in the same stratum for whom there are also measures of prior performance. Doing this allows us to approximate a controlled crossover experiment in which subjects are exposed to both the treatment and control and their outcomes are measured on each occasion. This experimental design is commonly employed in clinical and health research but relatively underused in social science.

After matching the challenges and identifying individuals with Topcoder rating who have at least one treatment and one control observation in one of the strata, we are left with 1,239 observations of 319 coders over 42 challenges in 12 strata; these coders made 12,914 successful code submissions altogether. Compared to the average contestant in the same challenge, our "subjects" are more experienced and higher rated; further, they make more submissions and perform better (Table 5). Nevertheless, in most cases, the observations in the matched data span the full range of values, giving us enough variability to make conclusions about a diverse set of individuals on Topcoder.

Statistical Models

Our data are partially crossed since individuals are nested within different challenges but not all individuals are present in all of the challenges. Essentially, we have a non-balanced design with non-independent observations and we account for it by fitting mixed effects models with random intercepts for strata, challenges, and individuals (Barr, Levy, Scheepers, & Tily, 2013). Since not all types of challenges are equally represented, we additionally weigh the observations by strata.

To test the hypotheses, the models include the coder's Topcoder rating Z_{ij}, which is a challenge-varying individual-level variable. Additionally, we control for treatment ordering effects by including the number of challenges the coder has already participated in, which is another challenge-varying individual-level covariate (P_{ij}). The following equation describes the general structure of our models:

$$Y_{ijk} = b_0 + K_{0k} + J_{0j} + I_{0i} + b_1 X_j + b_2 Z_{ij} + b_3 X_j Z_{ij} + b_4 P_{ij} + e_{ij}$$

where K_{0k} is the stratum random intercept, J_{0j} is the challenge random intercept, and I_{0i} is the individual random intercept. The interaction term between scoring type and coder rating $b_3 X_j Z_{ij}$ helps to disentangle the different effects we expect on high-performers and low-performers. In practice, we model the relationship between rating and proximity to rating as quadratic and the relationship between rating and code complexity, number of code lines, number of submissions, and time in challenge as cubic. This decision is informed by the shape of the bivariate relations depicted in Figure 1 and Figure 2 and the best model fit. These two criteria similarly inform our decisions about which terms to interact with scoring type. To test the hypotheses, we use the Chi-square difference test to compare the models with and without the direct and interaction effects that involve scoring type.

The outcome variables number of code lines written, number of submissions, and code complexity are strictly positive and can be seen as a continuous representation of count variables. The variables are heavily skewed to the right and hence, we fit gamma regression models with a log link function for these measures (Brooks et al., 2017; Faraway, 2005). The outcome variables proximity to rating and time in challenge are also positive but bounded by 1 and hence, we fit a beta regression model for these two cases. Since the beta distribution excludes the extreme values 0 and 1, we rescaled the two outcome variables using the equation $y' = (y(N-1) + 0.5)/N$, where N is the sample size. We use $N = 29,075$, which is the total number of challenge contestants (but notice, not unique coders) in our data (Smithson & Verkuilen, 2006).

Since we are using pruned data, we cannot claim to measure the treatment effect on the whole population. Rather, the estimand in our statistical models is the local average treatment effect for the treated (Iacus et al., 2012).

Results

To test the hypotheses causally, we restrict our analyses to individuals who participated in two or more challenges that differed in scoring type but were similar in every other aspect. We then use statistical

models to test whether the scoring type of the challenge affects the effort (H1), the quality of performance as measured by the cyclomatic complexity of the solution (H2), and the predictability of performance (H3) of individuals of different abilities in the predicted direction.

Effort

Compared to absolutely scored challenges, in relatively scored challenges low-rated individuals work less on their code, while high-rated individuals work more: on average, a contestant in the bottom 15% by rating writes and edits 46-245 fewer lines in relatively scored challenges compared to absolutely scored challenges while a contestant in the top 15% writes and edits 300-383 more (Fig. 1). This difference is statistically significant when we test it in regression models that account for variability by stratum, challenge, and individual, control for ordering effects, and weigh the observations by stratum ($\Delta\chi^2(2df) = 7.633$, $p = 0.022$ between Model 1.1 and Model 1.2 in Table 1). The negative effect of relative scoring on the effort exerted by low-rated individuals is also evident when we study how many submissions they made and how long they stayed active in the challenge. The difference for number of submissions is statistically significant ($\Delta\chi^2(2df) = 7.537$, $p = 0.023$ between Model 2.1 and 2.2 in Table 1), although not the one for duration ($\Delta\chi^2(2df) = 4.682$, $p = 0.096$ between Model 3.1 and 3.2 in Table 1). More concretely, Figure 1 shows that contestants in the bottom 45% by rating make on average 1.6-2.5 fewer submissions in relatively scored challenges compared to absolutely scored challenges but this difference starts to disappear for those with higher ratings. Overall, these results support H1. The evidence is strongest and most consistent when it comes to the negative effects of social comparison on the effort exerted by low-performing individuals.

Performance

We find that the difference in the level of performance, as measured by the computational complexity of the final submitted code, is more extreme between low-rated and high-rated individuals in challenges with relative scoring compared to challenges with absolute scoring (Fig. 2). The statistical models reveal that this difference is significant ($\Delta\chi^2(3df) = 11.532$, $p = 0.009$ between Models 4.1 and 4.2 in Table 2), providing support for H2. The effect is mainly driven by the higher code complexity produced by coders rated above the 50th percentile. The left panel in Figure 2 reveals that relatively scored challenges tend to result in more sophisticated code solutions, mainly because higher-rated individuals produce solutions with higher cyclomatic complexity: those in the top half by rating submit software with about 30-74 additional independent logical paths when scoring is relative. The lowest-rated individuals, in contrast, do not appear

Table 1: Results for effort.

	Total code lines		**Num. submissions**		**Time in challenge**	
	(1.1)	(1.2)	(2.1)	(2.2)	(3.1)	(3.2)
Relative scoring		**−0.269***		**−0.292****		**−0.341**
		(0.130)		**(0.109)**		**(0.194)**
Relative scoring × Rating		**0.405****		**0.338***		**0.531***
		(0.148)		**(0.140)**		**(0.247)**
Rating	5.340***	5.307***	4.109***	4.009***	6.898***	6.573***
	(1.046)	(1.045)	(1.007)	(1.007)	(1.557)	(1.561)
Rating^2	−8.260***	−8.675***	−6.834**	−7.058**	−10.520**	−10.540**
	(2.299)	(2.300)	(2.206)	(2.207)	(3.445)	(3.443)
Rating^3	4.673**	4.932***	3.879**	4.026**	5.641*	5.681**
	(1.467)	(1.467)	(1.415)	(1.416)	(2.204)	(2.204)
Number of prev. challenges	−0.002	−0.002	−0.004	−0.004	−0.001	−0.001
	(0.002)	(0.002)	(0.002)	(0.002)	(0.003)	(0.003)
Constant	5.476***	5.605***	1.292***	1.453***	−1.776***	−1.573***
	(0.155)	(0.170)	(0.172)	(0.181)	(0.232)	(0.258)
Observations	1025		1239		1229	
Individuals	279		319		317	
Challenges	40		42		42	
Strata	11		12		12	
Individual-level variance	0.223	0.221	0.300	0.301	0.344	0.346
Challenge-level variance	0.050	0.046	0.021	0.018	0.060	0.059
Stratum-level variance	0.027	0.027	0.116	0.110	0.092	0.083
Log Likelihood	−7821	−7817	−3847	−3843	614	616
AIC	15659	15656	7712	7708	−1209	−1210
$\Delta\chi^2$	**7.633* (2 df)**		**7.537* (2 df)**		**4.682 (2 df)**	

Note: Coefficients and standard errors (in brackets) for: (1) mixed effects gamma regression for the total number of code lines the coder wrote in the challenge; (2) mixed effects gamma regression for the number of submissions the coder made in the challenge; (3) mixed effects beta regression for the duration of the coder's participation in the challenge, measured as the difference between the time of their last and first submissions, divided by the duration of the challenge. The models include weights by stratum and control for treatment ordering effects using the number of previous challenges the coder has participated in. The models also include random intercepts by individual, challenge, and stratum.

* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$

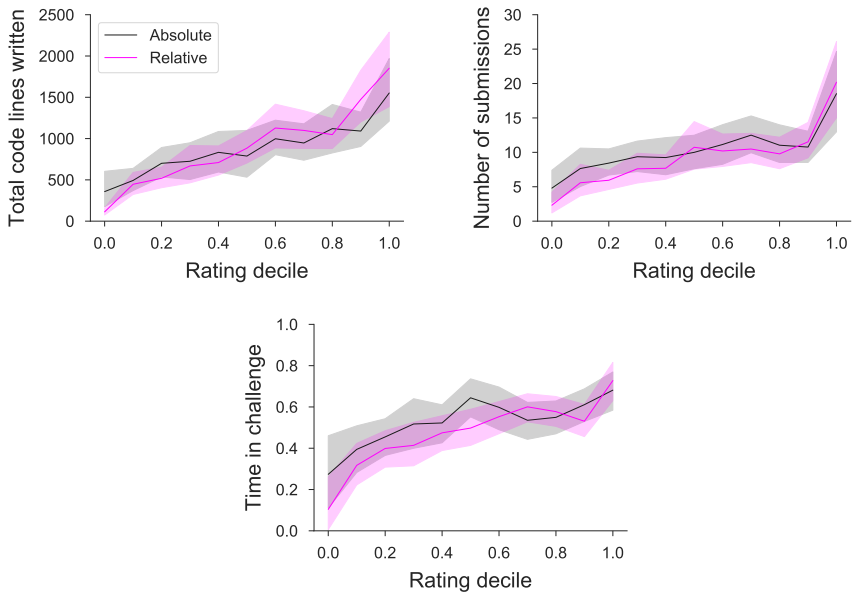


Figure 1: Low-rated coders put less effort in Topcoder challenges with relative scoring than in challenges with absolute scoring: they write fewer lines of code (top left), make fewer submissions (top right), and stay for shorter period out of the duration of the challenge (bottom). The shaded area indicates 95% confidence intervals around the means, which are estimated over 954 stratum-individuals.

to develop their solutions as much in relatively scored challenges.

Regarding the predictability of performance, the U-shaped relationship in the right panel in Figure 2 suggests a tendency for higher predictability at the lower and higher ends of rating. It appears that relatively scored challenges make the performance of low-rated individuals more predictable, while that of individuals in the 60-80 percentile by rating less predictable. This would suggest that relatively scored challenges are more likely to reinforce the position of low-rated coders but also more likely to produce upsets or surprise wins for above-average performers. This difference, however, does not reach statistical significance in our data and models: comparing the model without treatment effects (Model 5.1 in Table 2) and the model with treatment effects for the intercept and slope of the relationship between rating and predictability in the treated group (Model 5.2 in Table 2) yields $\Delta\chi^2(2df) = 3.974$ and $p = 0.137$. In short, the data do not provide sufficient evidence in support of H3.

Finally, to confirm that the differences in code complexity we observe in support of H2 result from the differential effect of scoring type on effort, we additionally compare the timing, length, and code complexity of the first submission (Fig. 5 and Table 6). The differences between contestants in the challenges with absolute and relative scoring are not statistically significant at the start of the challenges, indicating that the final outcomes are not driven by the contestants' *a priori* expectations and dispositions but are the result of social comparison dynamics.

Table 2: Results for performance.

	Code complexity		**Proximity to rating**	
	(4.1)	(4.2)	(5.1)	(5.2)
Relative scoring		**−0.189**		**0.201**
		(0.166)		**(0.108)**
Relative scoring × Rating		**1.480****		**−0.349***
		(0.544)		**(0.176)**
Relative scoring × Rating^2		**−1.171***		
		(0.495)		
Rating	4.354***	3.656***	−1.792***	−1.575***
	(0.882)	(0.176)	(0.384)	(0.397)
Rating^2	−6.801***	−6.477***	2.431***	2.420***
	(1.933)	(1.950)	(0.355)	(0.355)
Rating^3	4.043***	4.271***		
	(1.227)	(1.225)		
Num. previous challenges	−0.001	−0.001	0.004**	0.004*
	(0.002)	(0.002)	(0.002)	(0.002)
Constant	3.863***	3.946***	1.544***	1.426***
	0.139	(0.169)	(0.111)	(0.128)
Observations	1188		1239	
Individuals	308		319	
Challenges	42		42	
Strata	12		12	
Individual-level variance	0.138	0.136	0.028	0.028
Challenge-level variance	0.082	0.073	0.000	0.000
Stratum-level variance	0.031	0.030	0.002	0.002
Log Likelihood	−6752	−6747	1040	1042
AIC	13523	13517	−2063	−2063
$\Delta\chi^2$	**11.532**** **(3 df)**		**3.974 (2 df)**	

Note: Coefficients and standard errors (in brackets) for: (4) mixed effects gamma regression for the cyclomatic complexity of the code in the final submission; (5) mixed effects beta regression for the proximity between the coder's challenge-specific percentile rank by Topcoder rating and their percentile rank by final score in the challenge. The models include weights by stratum and control for treatment ordering effects using the number of previous challenges the coder has participated in. The models also include random intercepts by individual, challenge, and stratum.

* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$

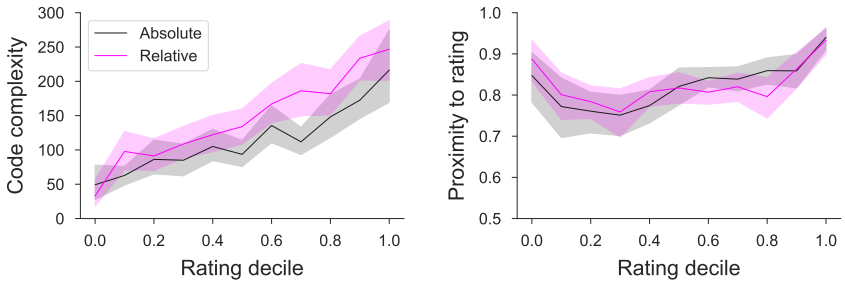


Figure 2: High-rated coders produce solutions with higher cyclomatic complexity compared to low-rated coders in Topcoder challenges with relative scoring than in challenges with absolute scoring (left). There is also indicative evidence that the performance of low-rated coders is more predictable in relatively scored challenges (right) but this result is not statistically significant (Table 2). The shaded area indicates 95% confidence intervals around the means, which are estimated over 954 stratum-individuals.

Discussion

Nowadays, some of the most common and prominent strategies to motivate and engage users on crowdsourcing platforms, social media networks, online learning portals, and gaming communities rely on the power of social comparison. Leaderboards, level achievements, badges, and prizes aim to stimulate the user's competitive drive and increase participation, effort, and creativity (Morschheuser, Hamari, Koivisto, & Maedche, 2017). These practical approaches are backed by decades of behavioral economics and organizational research on the positive effects of contests and tournaments on average individual performance. Nevertheless, researchers have also warned about possible backfire effects from social comparison and competition (Barker & Barclay, 2016; Charness et al., 2013; Liu et al., 2014), and the current study presents arguments and evidence aligning with this more measured and cautious outlook.

We hypothesized that feedback based on social comparison affects the effort of individuals of different ability differentially, in ways that make individual performance more predictable and unequal. We employed advanced computational measures and a crossover quasi-experiment to test these predictions with data from the online crowdsourcing community Topcoder. The empirical results support two of our three hypotheses. We found that when feedback for the submissions is based on relative performance, low-rated Topcoder participants reduce effort, while high-rated participants increase it. Although the quality of the submissions is not necessarily lower for the low-rated participants, it is markedly higher for the high-rated participants. Thus, as predicted, relative feedback leads to individual performance that is more extremely distributed in the group. Unfortunately, the data do not provide sufficient evidence for the prediction that performance becomes more predictable. Nevertheless, over time, the unequal effects on effort and performance we discovered could lead to a self-reinforcing loop that solidifies and even increases differences

between bottom and top performers.

Theoretical Implications

The study makes several important theoretical contributions. First, we enrich the economic and management literature on contests and tournaments by introducing the problem of inequality. Under the umbrella of "tournament theory," most prior research has focused on the optimal design of prize structures to increase average and best performance (Connelly et al., 2014; Lazear & Rosen, 1981). In contrast, we posit that the dispersion and predictability of effort and performance are just as important group outcomes to consider. If effort and performance reinforce pre-existing differences in skills and abilities, low-ability individuals may lose motivation to engage and learn. Thus, inequality in effort and performance has implications for the well-being, sustainability, and growth of organizations and communities, similarly to other manifestations of social and economic inequality (Bapuji et al., 2020).

Second, we contribute to social psychology theory by recognizing and partially testing a seminal hypothesis that has received little prior attention. Although indirectly, Festinger (Festinger, 1954) implied that social comparison should equalize individual outcomes because it makes underachievers exert more effort, receive more help, undermine overachievers, or altogether avoid tough competition. Our study bracketed self-selection and focused on how the same individuals change behavior under social comparison in non-cooperative situations. Thus, we directly addressed Festinger's prediction on effort and proved it wrong in competitive settings. We found that, in the absence of opportunities for cooperation, social comparison can actually make underachievers exert less effort, not more, and thus produce more unequal performance. Our findings pave the way for future theoretical and empirical work that comprehensively revisits, revises, and restricts Festinger's predictions on the group effects of social comparison.

Last but not least, we add to the literatures on crowdsourced contests, contest design, and gamified crowdsourcing (Adamczyk et al., 2012; Feng, Jonathan Ye, Yu, Yang, & Cui, 2018; Morschheuser et al., 2017) by shedding light on the differential effects that feedback based on social comparison can have on users of different ability. From previous empirical research on the Topcoder community we know that high-skilled contestants affect contest dynamics by discouraging entry (Archak, 2010) and eliciting higher effort and greater performance from other top performers (K. J. Boudreau et al., 2016); here, we expand this view by including the converse effects on low performers too. In fact, prior research on gamification has already raised an alarm that long-term leaderboards may lead to inequality (Preist, Massung, & Coyle, 2014). Specifically, one experimental study of dyadic tournaments on an online labor market shows

that while leaderboards make weak participants who face strong competitors drop out, they make strong participants who face strong competitors compete harder (Straub, Gimpel, Teschner, & Weinhardt, 2015). We extend this work by studying tournaments with more complex and realistic tasks, longer duration, and more participants, while retaining focus on causal inference. Overall, our argument that the exploitation of social comparison for competition may have undesirable long-term and group-level consequences aligns well with the recent shift in innovation contest design towards interaction, cooperation, and idea sharing (Bailey & Horvitz, 2010; Bullinger et al., 2010; LaToza et al., 2015; Malone et al., 2017; McInnis, Xu, & Dow, 2018).

Practical Implications

Our findings suggest that relying on social comparison for performance feedback could negatively affect underachievers in competitive settings. Although Topcoder relies on commercial programming challenges to finance itself, most of the challenges it organizes are designed and organized with the sole purpose to stimulate and engage the Topcoder community. In fact, our results are based precisely on the latter type of challenges, which are not monetarily rewarded. Thus, the direct implication from our study is that Topcoder, and other similar contest-based online crowdsourcing communities, should carefully consider whether to use evaluation and feedback focusing on relative performance if the aim is to motivate and engage less active and less invested users. While relative feedback may stimulate experienced overachievers, it may demotivate learners. Crowdsourcing platform providers should stay acutely aware of this tradeoff when it comes to individuals competing against each other.

The focus of our study also brings attention to the need to consider long-term group-level outcomes in addition to short-term goals and effects when designing crowd-based contests. Leaderboards, rankings, and other-based feedback may help elicit value for clients in the short term but may also drive away some users, inadvertently impacting the size, composition, and identity of the community. Yet, it is not social comparison per se that is the problem but its use to heighten competition. When combined with opportunities for cooperation and idea exchange, social comparison can in fact foster learning, community building, and innovation. For example, innovation contests that make submissions public and allow for peer comments and evaluation help contestants learn and incorporate ideas from others' solutions and improve the quality of their own solution (Bockstedt et al., 2016; Bullinger et al., 2010). Importantly, low and top performers appear to benefit similarly from such an arrangement (LaToza et al., 2015).

Beyond crowdsourcing, the study carries implications more generally for computer-supported commu-

nities and organizations that are centered around the learning, engagement, and continuous participation of their members. For example, there is a trend to use leaderboards and badges to gamify learning experiences but, complementing other empirical research (Domínguez et al., 2013; Hanus & Fox, 2015), our findings suggest that this innovation may not be equally beneficial for students of all ability levels if the emphasis is on individual competition. "Grading on a curve", whereby course grades are determined in reference to others' performance, has been shown to increase average effort under some conditions (Andreoni & Brownback, 2017; Becker & Rosen, 1992) but our research suggests that it might also increase the disparity in effort and performance between bottom achievers and top achievers. Along the same lines, some organizations employ a performance management practice known as "stack ranking", "forced ranking", or "vitality curve", whereby employees are rated against their coworkers (Stewart, Gruys, & Storm, 2010). Our research suggests that even if such performance feedback is not directly related to financial rewards and promotion, it can trigger a self-reinforcing process and lower effort further among bottom performers.

Limitations and Future Research

Despite the advanced computational measures and novel causal inference methods employed by the study, several limitations remain. To provide more accurate causal estimates, we used matching and a quasi-experimental crossover design. However, this made us disregard the majority of our data and narrow down the scope of our claims. For example, our findings do not necessarily transfer to contests that offer monetary rewards. In those cases, financial considerations could overwhelm the subtle behavioral effects from social comparison we found but could also possibly interact with them (Cerasoli, Nicklin, & Ford, 2014; Liang, Wang, Wang, & Xue, 2018; Rogstadius et al., 2011).

We also recognize that our sophisticated matching techniques do not entirely preclude the possibility for confounding. Despite matching on multiple aspects, including task difficulty, task type, task goal, and the complexity of the scoring procedure, it is still possible that relatively and absolutely scored challenges differ in respects that we have not captured and that may explain the observed effects. This is an inherent limitation to any causal inference research that relies on observational data such as ours. We assumed that the only difference between the two types of challenges is the emphasis on social comparison but we cannot entirely preclude other subtle differences responsible for alternative behavioral mechanisms. For example, it is possible that relative scoring is mainly applied to challenges that are more open-ended and less structured and it is in fact the absence of clear detailed instructions that leads low performers to lose motivation and top performers to gain ambition. Ideally, we would have survey data on how coders

experienced the challenge and what motivated their contribution. Even better, we would randomly assign coders of varying abilities to identical challenges that only differ on the scoring procedure. As always, randomized field or lab experiments are the gold standard for causal claims, whenever they are possible and feasible.

The measure for the quality of performance we employed—the cyclomatic complexity of the final submitted code—has limitations too. We required an objective measure of performance that is not correlated with the scoring type; we thus could not rely on the official performance metrics employed by the challenges. Further, the measure had to work for multiple programming languages and be easily calculated for thousands of code submissions, each involving hundreds of code lines. Evaluating the quality of software code is a difficult task and there is a proliferation of metrics available (Munson & Khoshgoftaar, 1989; Fenton & Neil, 2000). We chose McCabe's cyclomatic complexity with the intention to quantify software quality in terms of its elaborated logical flow and development. Although theoretically simple and appealing, empirically, the metric has been shown to be strongly correlated with lines of code (Shepperd, 1988). Indeed, the Pearson correlation between the two for the final code submissions we analyze is 0.878. We thus caution that our results on the quality of performance should be interpreted more narrowly in terms of the extent of elaboration and development of the solution, and not in reference to other possible criteria such as software maintainability, optimization, code legibility, etc.

Further, our research design is powerful when it comes to dealing with self-selection bias, but has limitations with respect to sample selection bias. We know for a fact that the matched sample we study does not represent the Topcoder population accurately. In supplementary analyses, we found out that our results concern only coders with prior experience on the platform and do not extend to first-time contestants. Assuming that first-timers randomly choose their first challenge, we tested if the performance, effort, and likelihood to participate again are different for those who happened to enter Topcoder with a relatively scored challenge versus absolutely scored challenge. However, the only statistically significant difference is that first-timers in relatively scored challenges tend to exit the contest faster than first-timers in absolutely scored challenges (Table 7). One possibility is that the effects on newcomers are exactly the same as for experienced Topcoder users but since we don't observe their ability, we cannot measure them. Of course, another possibility is that the effects do not extend to naïve participants and thus, our findings are restricted to a specific context and population.

We recognize that the Topcoder community is not representative of other crowdsourcing platforms or online communities in general. Thus, there is an obvious need to replicate our findings on other online

sites and in other contexts. This will help clarify the frequency, strength, and conditions for the effects we observed. Specifically, we note that although we framed our argument more broadly in terms of social comparison, the effects could deffer for upward and downward social comparison. Out of the 31 relatively scored challenges we analyze, in six the score depends on the number of competitors the player beats (downward comparison) while in the rest, the score is normalized by the current best score (upward comparison). When we split the analyses by the direction of comparison, we observe that downward comparison appears to be the one that demotivates those already at the bottom of the ranking, while upward comparison is the one that motivates those at the top (Fig. 6). Because we halve the data after the split, we cannot disentangle these effects quantitatively but we hope future research can test for this possible effect difference. Another hypothesis is that the negative effects of social comparison on under-achievers may be non-existent, or even positive, in smaller, less anonymous, and more collaboration-centered groups. If proven, such result will delineate more precisely the scope of Festinger's arguments (Festinger, 1954), as well as the scope of our findings here.

Conclusion

To motivate user effort and engagement, many online platforms introduce game elements such as rankings, leaderboards, and contests (Deterding, Dixon, Khaled, & Nacke, 2011; Morschheuser et al., 2017). These elements rely on humans' natural proclivity for social comparison and status seeking to trigger competition and interest (Festinger, 1954; Garcia et al., 2013). However, using data from the online crowdsourcing platform Topcoder, we presented evidence that social comparison increases effort for better-performing individuals but has exactly the opposite effects on worse-performing individuals. This can trigger a rich-get-richer-poor-get-poorer dynamic that increases engagement and performance differences in the group even further. In this way, social comparison and competition can negatively impact motivation and learning, and organizations and communities that focus on equity and sustainability should be careful to promote them.

Acknowledgements

M.T., S.M., and O.V. acknowledge the generous support of the Volkswagen Foundation under Grant Ref. 92 173. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. R.S. is funded by the Division of Research and Faculty Development at

Harvard Business School.

References

Adamczyk, S., Bullinger, A. C., & Möslein, K. M. (2012). Innovation contests: A review, classification and outlook. *Creativity and Innovation Management*, *21*(4), 335–360. doi: 10.1111/caim.12003

Allison, P. D. (1978). Measures of inequality. *American Sociological Review*, *43*(6), 865–880. doi: 10.2307/2094626

Andreoni, J., & Brownback, A. (2017). All pay auctions and group size: Grading on a curve and other applications. *Journal of Economic Behavior & Organization*, *137*, 361–373.

Archak, N. (2010). Money, glory and cheap talk: Analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder.com. In *Proceedings of the 19th International Conference on World Wide Web* (pp. 21–30). New York, NY, USA: ACM. (event-place: Raleigh, North Carolina, USA)

Archak, N., & Sundararajan, A. (2009). Optimal design of crowdsourcing contests. *ICIS 2009 Proceedings*, *200*, 1–16.

Bailey, B. P., & Horvitz, E. (2010). What's your idea? a case study of a grassroots innovation pipeline within a large software company. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2065–2074). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/1753326.1753641

Bapuji, H., Ertug, G., & Shaw, J. D. (2020). Organizations and societal economic inequality: A review and way forward. *Academy of Management Annals*, *14*(1), 60–91.

Barclay, P., & Stoller, B. (2014, May). Local competition sparks concerns for fairness in the ultimatum game. *Biology Letters*, *10*(5), 20140213.

Barker, J. L., & Barclay, P. (2016). Local competition increases people's willingness to harm others. *Evolution and Human Behavior*, *37*(4), 315–322.

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, *68*(3), 255–278.

Becker, W. E., & Rosen, S. (1992). The learning effect of assessment and evaluation in high school. *Economics of Education Review*, *11*(2), 107–118.

Bockstedt, J., Druehl, C., & Mishra, A. (2016). Heterogeneous submission behavior and its implications for success in innovation contests with public submissions. *Production and Operations Management*,

25(7), 1157–1176. doi: 10.1111/poms.12552

Boudreau, K., Helfat, C. E., Lakhani, K. R., & Menietti, M. E. (2012). Field evidence on individual behavior and performance in rank-order tournaments. *Harvard Business School Working Paper Series # 13-016*, *13-016*, 1–36.

Boudreau, K. J., Lacetera, N., & Lakhani, K. R. (2011, April). Incentives and problem uncertainty in innovation contests: An empirical analysis. *Management Science*, *57*(5), 843–863.

Boudreau, K. J., Lakhani, K. R., & Menietti, M. (2016). Performance responses to competition across skill levels in rank-order tournaments: field evidence and implications for tournament design. *The RAND Journal of Economics*, *47*(1), 140–165. doi: 10.1111/1756-2171.12121

Brooks, M. E., Kristensen, K., van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., ... Bolker, B. M. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R Journal*, *9*(2), 378–400.

Bullinger, A. C., Neyer, A.-K., Rass, M., & Moeslein, K. M. (2010). Community-based innovation contests: Where competition meets cooperation. *Creativity and Innovation Management*, *19*(3), 290–303. doi: 10.1111/j.1467-8691.2010.00565.x

Buunk, B. P., Collins, R. L., Taylor, S. E., VanYperen, N. W., & Dakof, G. A. (1990). The affective consequences of social comparison: Either direction has its ups and downs. *Journal of Personality and Social Psychology*, *59*(6), 1238–1249. doi: 10.1037/0022-3514.59.6.1238

Callan, M. J., Shead, N. W., & Olson, J. M. (2011). Personal relative deprivation, delay discounting, and gambling. *Journal of Personality and Social Psychology*, *101*(5), 955–973.

Cason, T. N., Masters, W. A., & Sheremeta, R. M. (2010, October). Entry into winner-take-all and proportional-prize contests: An experimental study. *Journal of Public Economics*, *94*(9), 604–611.

Cerasoli, C. P., Nicklin, J. M., & Ford, M. T. (2014). Intrinsic motivation and extrinsic incentives jointly predict performance: A 40-year meta-analysis. *Psychological Bulletin*, *140*(4), 980–1008. doi: 10.1037/a0035661

Charness, G., Masclet, D., & Villeval, M. C. (2013, August). The dark side of competition for status. *Management Science*, *60*(1), 38–55.

Chen, K.-P. (2003, April). Sabotage in promotion tournaments. *The Journal of Law, Economics, and Organization*, *19*(1), 119–140.

Chowdhury, S. M., & Gürtler, O. (2015, July). Sabotage in contests: A survey. *Public Choice*, *164*(1-2), 135–155.

Collins, R. L. (1996). For better or worse: The impact of upward social comparison on self-evaluations. *Psychological Bulletin*, *119*(1), 51–69. doi: 10.1037/0033-2909.119.1.51

Connelly, B. L., Tihanyi, L., Crook, T. R., & Gangloff, K. A. (2014, January). Tournament theory: Thirty years of contests and competitions. *Journal of Management*, *40*(1), 16–47.

Cook, K. S., & Hegtvedt, K. A. (1983). Distributive justice, equity, and equality. *Annual Review of Sociology*, *9*(1), 217–241. doi: 10.1146/annurev.so.09.080183.001245

Dechenaux, E., Kovenock, D., & Sheremeta, R. M. (2015). A survey of experimental research on contests, all-pay auctions and tournaments. *Experimental Economics*, *18*(4), 609–669.

Denny, P., McDonald, F., Empson, R., Kelly, P., & Petersen, A. (2018, April). Empirical support for a causal relationship between gamification and learning outcomes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (pp. 1–13). New York, NY, USA: Association for Computing Machinery.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 9–15). New York, NY, USA: ACM. doi: 10.1145/2181037.2181040

Dickinson, D. L., & Isaac, R. M. (1998). Absolute and relative rewards for individuals in team production. *Managerial and Decision Economics*, *19*(4/5), 299–310.

DiPalantino, D., & Vojnovic, M. (2009). Crowdsourcing and all-pay auctions. In *Proceedings of the 10th ACM Conference on Electronic Commerce* (pp. 119–128). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/1566374.1566392

Dohmen, T., & Falk, A. (2011, April). Performance pay and multidimensional sorting: Productivity, preferences, and gender. *American Economic Review*, *101*(2), 556–590.

Domínguez, A., Saenz-de Navarrete, J., de Marcos, L., Fernández-Sanz, L., Pagés, C., & Martínez-Herráiz, J.-J. (2013). Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*, *63*, 380–392.

Eriksson, T., Teyssier, S., & Villeval, M.-C. (2009). Self-selection and the efficiency of tournaments. *Economic Inquiry*, *47*(3), 530–548. doi: 10.1111/j.1465-7295.2007.00094.x

Faraway, J. J. (2005). *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models* (1edition ed.). Boca Raton: Chapman and Hall/CRC.

Feng, Y., Jonathan Ye, H., Yu, Y., Yang, C., & Cui, T. (2018). Gamification artifacts and crowdsourcing

participation: Examining the mediating role of intrinsic motivations. *Computers in Human Behavior*, *81*, 124–136. doi: 10.1016/j.chb.2017.12.018

Fenton, N. E., & Neil, M. (2000). Software metrics: roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (pp. 357–370). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/336512.336588

Festinger, L. (1954, May). A theory of social comparison processes. *Human Relations*, *7*(2), 117–140.

Fiske, S. T. (2011). *Envy Up, Scorn Down: How Status Divides Us.* New York, NY: Russell Sage Foundation.

Garcia, S. M., Tor, A., & Gonzalez, R. (2006). Ranks and Rivals: A Theory of Competition. *Personality and Social Psychology Bulletin*, *32*(7), 970–982.

Garcia, S. M., Tor, A., & Schiff, T. M. (2013). The psychology of competition: A social comparison perspective. *Perspectives on Psychological Science*, *8*(6), 634–650.

Hanus, M. D., & Fox, J. (2015). Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & Education*, *80*, 152–161.

Harbring, C., & Irlenbusch, B. (2011, March). Sabotage in tournaments: Evidence from a laboratory experiment. *Management Science*, *57*(4), 611–627.

Howe, J. (2008). *Crowdsourcing: How the Power of the Crowd is Driving the Future of Business.* London, UK: Random House.

Hunt, J. W., & MacIlroy, M. D. (1976). *An Algorithm for Differential File Comparison.* Murray Hill, NJ: Bell Laboratories.

Iacus, S. M., King, G., & Porro, G. (2012). Causal inference without balance checking: Coarsened exact matching. *Political Analysis*, *20*(1), 1–24.

Jasso, G. (1983). Fairness of individual rewards and fairness of the reward distribution: Specifying the inconsistency between the micro and macro principles of justice. *Social Psychology Quarterly*, *46*(3), 185–199. doi: 10.2307/3033790

Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., ... Horton, J. (2013, February). The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work* (pp. 1301–1318). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2441776.2441923

Koivisto, J., & Hamari, J. (2019, April). The rise of motivational information systems: A review

of gamification research. *International Journal of Information Management*, *45*, 191–210. Retrieved from https://www.sciencedirect.com/science/article/pii/S0268401217305169 doi: 10.1016/j.ijinfomgt.2018.10.013

Lakhani, K. R., Boudreau, K. J., Loh, P.-R., Backstrom, L., Baldwin, C., Lonstein, E., ... Guinan, E. C. (2013, February). Prize-based contests can provide solutions to computational biology problems. *Nature Biotechnology*, *31*(2), 108–111.

Landers, R. N., Bauer, K. N., & Callan, R. C. (2017). Gamification of task performance with leaderboards: A goal setting experiment. *Computers in Human Behavior*, *71*, 508–515.

LaToza, T. D., Chen, M., Jiang, L., Zhao, M., & Hoek, A. v. d. (2015). Borrowing from the crowd: A study of recombination in software design competitions. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 1, pp. 551–562). Florence, Italy: IEEE. doi: 10.1109/ICSE.2015.72

Lazear, E. P., & Rosen, S. (1981, October). Rank-order tournaments as optimum labor contracts. *Journal of Political Economy*, *89*(5), 841–864.

Liang, H., Wang, M.-M., Wang, J.-J., & Xue, Y. (2018). How intrinsic motivation and extrinsic incentives affect task effort in crowdsourcing contests: A mediated moderation model. *Computers in Human Behavior*, *81*, 168–176. doi: 10.1016/j.chb.2017.11.040

Liu, G., Lin, C., & Xin, Z. (2014, July). The effects of within- and between-group competition on trust and trustworthiness among acquaintances. *PLOS ONE*, *9*(7), e103074.

Lynn, F., Podolny, J. M., & Tao, L. (2009). A sociological (de)construction of the relationship between status and quality. *American Journal of Sociology*, *115*(3), 755–804. doi: 10.1086/603537

Mago, S. D., Samak, A. C., & Sheremeta, R. M. (2016, April). Facing your opponents: Social identification and information feedback in contests. *Journal of Conflict Resolution*, *60*(3), 459–481.

Malone, T. W., Nickerson, J. V., Laubacher, R. J., Fisher, L. H., de Boer, P., Han, Y., & Towne, W. B. (2017). Putting the pieces back together again: Contest webs for large-scale problem solving. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (pp. 1661–1674). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2998181.2998343

McCabe, T. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, *SE-2*(4), 308–320.

McInnis, B., Xu, X. T., & Dow, S. P. (2018). How features of a civic design competition influences the

collective understanding of a problem. *Proc. ACM Hum.-Comput. Interact.*, *2*(CSCW), 120:1–120:25. doi: 10.1145/3274389

Messick, D. M., & McClintock, C. G. (1968). Motivational bases of choice in experimental games. *Journal of Experimental Social Psychology*, *4*(1), 1–25.

Morschheuser, B., Hamari, J., Koivisto, J., & Maedche, A. (2017). Gamified crowdsourcing: Conceptualization, literature review, and future agenda. *International Journal of Human-Computer Studies*, *106*, 26–43.

Munson, J., & Khoshgoftaar, T. (1989). The dimensionality of program complexity. In *11th International Conference on Software Engineering* (pp. 245–253). Pittsburgh, PA, USA: IEEE. doi: 10.1109/ICSE.1989.714426

Payne, B. K. (2017). *The Broken Ladder: How Inequality Affects the Way We Think, Live, and Die*. New York: Viking.

Piketty, T., & Goldhammer, A. (2014). *Capital in the Twenty-First Century*. Cambridge Massachusetts: Harvard University Press.

Preist, C., Massung, E., & Coyle, D. (2014). Competing or aiming to be average? normification as a means of engaging digital volunteers. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing* (pp. 1222–1233). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2531602.2531615

Ratcliff, J. W., & Metzener, D. E. (1988). Pattern-matching-the gestalt approach. *Dr Dobbs Journal*, *13*(7), 46.

Restivo, M., & van de Rijt, A. (2012). Experimental study of informal rewards in peer production. *PLoS ONE*, *7*(3), e34358. doi: 10.1371/journal.pone.0034358

Rogstadius, J., Kostakos, V., Kittur, A., Smus, B., Laredo, J., & Vukovic, M. (2011). An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In *Fifth International AAAI Conference on Weblogs and Social Media* (pp. 321–328). Palo Alto, CA: Association for the Advancement of Artificial Intelligence.

Salganik, M. J., Dodds, P. S., & Watts, D. J. (2006). Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, *311*(5762), 854–856. doi: 10.1126/science.1121066

Shepperd, M. (1988). A critique of cyclomatic complexity as a software metric. *Software Engineering Journal*, *3*(2), 30–36. doi: 10.1049/sej.1988.0003

Sheremeta, R. M. (2010, March). Experimental comparison of multi-stage and one-stage contests. *Games*

and Economic Behavior, *68*(2), 731–747.

Smithson, M., & Verkuilen, J. (2006). A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. *Psychological Methods*, *11*(1), 54–71.

Stainback, K., Tomaskovic-Devey, D., & Skaggs, S. (2010). Organizational Approaches to Inequality: Inertia, Relative Power, and Environments. *Annual Review of Sociology*, *36*(1), 225–247.

Stewart, S. M., Gruys, M. L., & Storm, M. (2010). Forced distribution performance evaluation systems: Advantages, disadvantages and keys to implementation. *Journal of Management & Organization*, *16*(1), 168–179.

Straub, T., Gimpel, H., Teschner, F., & Weinhardt, C. (2015). How (not) to incent crowd workers. *Business & Information Systems*, *57*(3), 167–179. doi: 10.1007/s12599-015-0384-2

Travis, J. (2008, March). Science by the masses. *Science*, *319*(5871), 1750–1752.

Tsvetkova, M., Müller, S., Vuculescu, O., Ham, H., & Sergeev, R. (2022). *Data and software for "relative feedback increases disparities in effort and performance in crowdsourcing contests".* figshare. doi: 10.6084/m9.figshare.20292279.v1

Van de Werfhorst, H. G., & Mijs, J. J. (2010). Achievement Inequality and the Institutional Structure of Educational Systems: A Comparative Perspective. *Annual Review of Sociology*, *36*(1), 407–428.

van Dijk, F., Sonnemans, J., & van Winden, F. (2001). Incentive systems in a real effort experiment. *European Economic Review*, *45*(2), 187–214. doi: 10.1016/S0014-2921(00)00056-8

West, S. A., Gardner, A., Shuker, D. M., Reynolds, T., Burton-Chellow, M., Sykes, E. M., ... Griffin, A. S. (2006). Cooperation and the scale of competition in humans. *Current Biology*, *16*(11), 1103–1106.

Wright, B. D. (1983). The economics of invention incentives: Patents, prizes, and research contracts. *American Economic Review*, *73*(4), 691–707.

Yin, T. (2019). *Lizard: A simple code complexity analyser. Python package version 1.16.6.* https://github.com/terryyin/lizard. Retrieved 2019-11-07, from `https://github.com/terryyin/lizard` (Retrieved from: https://github.com/terryyin/lizard)

Appendix

Data

Data Collection

For each challenge, we obtained data from Topcoder on the number of registrants, the number of contestants, an estimate of the duration of the challenge, and an estimate of the average rating of the contestants based on their performance in prior challenges. In addition, we have manual classifications of the challenges along a number of less straightforward categories—for example, ease of entry, the complexity of the scoring function, the type of the task, and the type of scoring. These classifications were computed from answers provided by coders in a dedicated challenge on Topcoder itself (`https://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=17094&pm=14852`).

For each contestant in a challenge, we know their Topcoder rating, the number of challenges in which they participated before, and their final score and rank in the challenge. To calculate a coder's rating, Topcoder uses a complex formula that considers the coder's actual performance in reference to their expected performance given all contestants' previous rating, the volatility of performance, and the number of previous contests (`https://community.topcoder.com/longcontest/?module=Static&d1=support&d2=ratings`).

Additionally, for each submission the contestant made in the challenge, we have the actual submitted code and know the time of submission, the programming language, and the provisional score achieved.

Data Cleaning

Each challenge poses a well-defined problem and has a unique leaderboard and prize structure. However, sometimes different challenges may be based on the same problem or challenges may be complementary, with each of them covering one aspect of a larger problem, often with an increasing level of difficulty. To avoid outlier effects in our analyses, we excluded 111 challenges that were based on a single problem (these presented a controlled experiment on the platform), leaving us with 256 challenges based on 214 unique problems. We additionally removed any challenge that had fewer than 10 contestants (5 challenges). The final set of 52 matched challenges corresponds to 49 unique problems.

When analyzing individual submissions we ignored 0-point submissions. These occur extremely often in the data and could be explained with minor bugs in the code that break the provisional scoring system. They are thus not indicative of effort in the same way that a new code revision is.

Similarly, when analyzing contestants in a challenge, we only considered individuals who made at least one non-zero submission. Although any coder who made a submission, whether successful or not, was considered a contestant and attributed a new rating as a result of their rank in the challenge, it is impossible to discern whether coders with only 0-point submissions made consistent effort or simply tested the system without being aware of the repercussions. Consistent with our decision to ignore 0-point submissions, we considered that these individuals made 0 successful submissions and, in effect, did not participate.

New contestants who have just joined the platform by definition lack a Topcoder rating. Since challenges that took place soon after Topcoder started contain mainly new platform members, the number of observations that we end up analyzing when rating is involved is actually lower: 42 instead of the 52 challenges we originally matched. The analyses involving cyclomatic complexity and file diffs also contain a number of missing values due to the algorithms failing to run on the submitted code.

Measures

Scoring Type

The classification by scoring type was provided by Topcoder users in a dedicated challenge on the platform. They were asked to mark the scoring type as absolute when "The score shows absolute performance of the solution on the test" and as relative when "The score shows performance relative to the performance of the other solutions". Below we provide two examples of how the scoring description might look like in each case.

> **Stratum 5, Absolute scoring:** "You will score 100 points for each coal that is dumped into a shaft. 1 point is deducted for each time step that you use, this relates to the running costs of your trucks. You may only use a maximum of 10000 time steps. All time steps greater than 10000 will be ignored. Your score for a test case will be Max(0, 100*(Coal gathered) - (Simulation steps used)). Any invalid move will result in a zero score. Your overall score will be the sum of your scores over all test cases."

> **Stratum 5, Relative scoring:** "Your score for an individual test case will be the BEST/YOU. Where YOU is the number of moves returned by your solution and BEST is the lowest number of moves returned by any of the competitors. Any kind of failure (invalid return, exceeding time/memory limit, moves not resulting in connected groups) will result in 0 score for that test. Your total score is simply the sum of scores for every test case."

Stratum 12, Absolute scoring: "The score for a test case will be 100 * max(0, 1 - (number of shifts in your return) / (SZ*SZ)). Invalid returns or returns which result in non-connected group of white tiles result in 0 score for that test case. The overall score is calculated as a sum of individual scores for all test cases."

Stratum 12, Relative scoring: "Your score for an individual test case will be the sum of sizes of your returned polygons. If your return has invalid format or specifies any invalid polygons, your score for the test case will be 0. Your overall score will be calculated in the following way: for each test case where your score is not 0, you get 1 point for each competitor you beat on this test case (i.e., your score on a test case is larger than this competitor's score) and 0.5 points for each competitor you tie with (a tie with yourself is not counted); finally, the sum of points is divided by (the number of competitors - 1). "

Code Complexity

The complexity of the source code was measured using cyclomatic complexity, as introduced by Thomas J. McCabe (McCabe, 1976). This software complexity measure uses the control graph of the code in order to determine the maximum number of linearly independent paths through the code. It is based on the idea that the complexity of the code is independent of its length and can be measured instead with the number of decision points.

We calculated cyclomatic complexity with the Python package Lizard (Yin, 2019). Lizard is a cyclomatic complexity analyzer that can be applied to all of the major programming languages used in the Topcoder source code files, including C++, C#, Java, and Python. The software increments the complexity measure by one unit each time it finds a condition in a function. For example, the Python conditions that the algorithm looks for are "if", "for", "while", "and", "or", "elif", "except", and "finally". Each of these conditions marks the creation of another linearly independent path through the code and therefore increases the complexity by one. The scores are calculated for each function in the source code and then summed to give the final measure.

Total Number of Code Lines Written

To quantify effort, we estimate the total number of lines the coder wrote for the challenge. To do this, we sum over the number of new code lines inserted with every submission. We first select all submissions that received a non-zero score (solutions which fail to compile receive a score of zero). We then clean up all

comments in the code files, alongside trailing and leading spaces. Finally, we use a matlab implementation of the diff algorithm to count the insertions in each submission compared to the previous one. The diff utility is a common tool to compare two sets of data (e.g. files) line by line (Hunt & MacIlroy, 1976). This differentiates it, for example, from the edit distance, another popular text comparison measure, which executes the comparison by character. The particular implementation of the diff utility we used (SimpleDiff) was developed by Paul Butler and relies on the Ratcliff/Obershelp algorithm to solve the longest common subsequence problem in order to determine the smallest set of deletions and insertions to create one set from the other (Ratcliff & Metzener, 1988).

Supplementary Figures and Tables

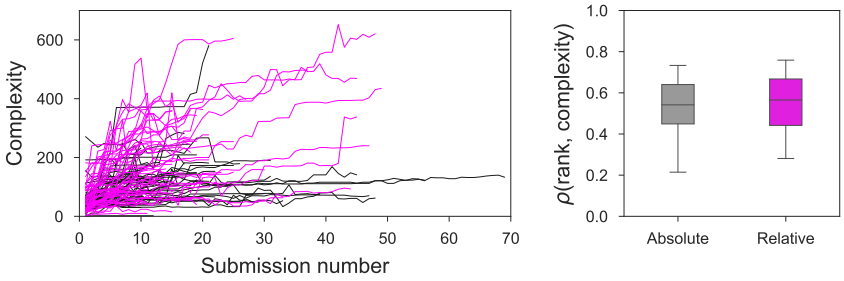


Figure 3: The cyclomatic complexity of the source code is a suitable measure of performance because it generally increases in subsequent submissions (left) and is correlated with rank percentile by final score in the challenge (right). The left plot shows the cyclomatic complexity of the submitted code for each subsequent submission for 100 contestants who made at least 10 submissions, randomly selected from the sample of matched individuals. The right plot shows boxplots of the Spearman rank correlation between the cyclomatic complexity of the code in the final submission and the rank percentile by final score in the 52 matched challenges.

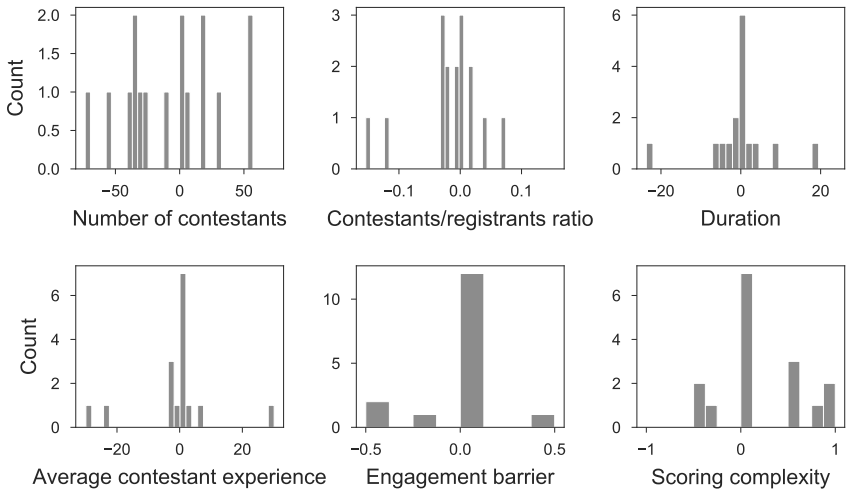


Figure 4: Covariate balance for the coarsened variables used for the coarsened exact matching. The figure shows histograms of the difference in means between the relatively scored and the absolutely scored challenges in the 16 matched strata. A positive value means that the relatively scored challenges in the particular stratum have a higher value for this covariate than the absolutely scored challenges in that stratum; the converse holds for a negative value.

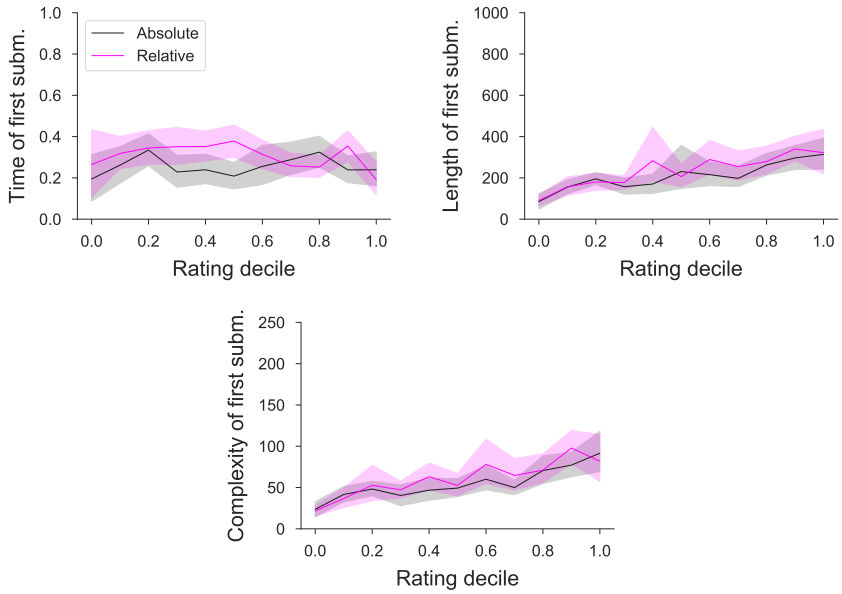


Figure 5: There are no significant differences in the timing (top left), length (top right), and complexity (bottom) of the first submission between relatively scored and absolutely scored challenges. Error bars and shaded areas indicate 95% confidence intervals around the means, which are estimated over 954 stratum-individuals.

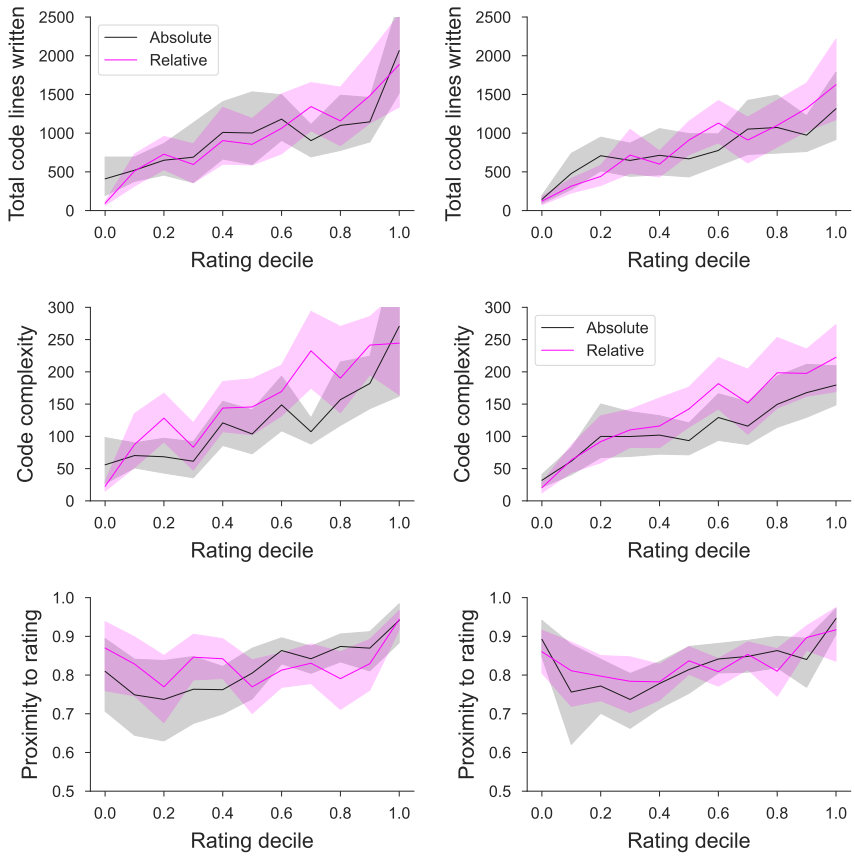


Figure 6: Low-rated coders put less effort and produce solutions with lower cyclomatic complexity in relatively scored challenges with downward social comparison (left), while high-rated coders put more effort and produce solutions with higher cyclomatic complexity in relatively scored challenges with upward social comparison (right). Error bars and shaded areas indicate 95% confidence intervals around the means, which are estimated over 450 (left) and 486 (right) stratum-individuals.

Table 3: Description of the challenge variables used for the coarsened exact matching ($N = 256$).

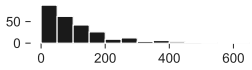
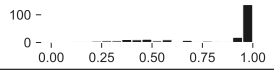
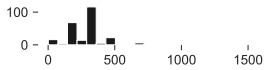
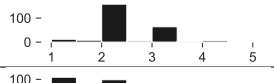
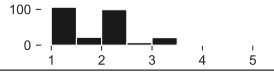
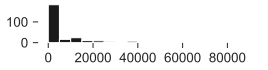
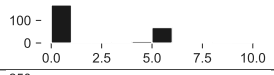
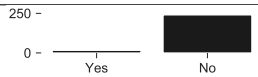
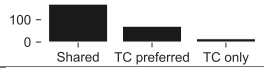
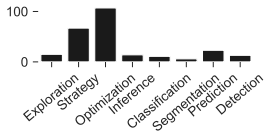
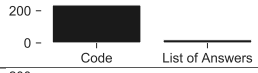
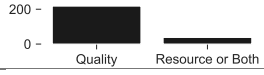
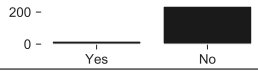
Variable name	Distribution	Coarsening
Scoring type		
Number of contestants		Less than 50 50–149 150–299 300 or more
Contestants/registrants ratio		[0, 0.5) [0.5, 0.9) [0.9, 1]
Duration		8–72 hrs (0.3–3 days) 108–204 hrs (4.5–8.5 days) 228–504 hrs (9.5–21 days) 600–1512 hrs (25–63 days)
Average contestant experience		Bottom tercile Mid tercile Top tercile
Engagement barrier		[1, 2] [2.5, 4]
Scoring complexity		[1, 2] [2.5, 3]
Prize pool		None $1,000–$7,500 $10,000–$30,000 $35,000 or more
Number of prizes		0 1–5 8–10
Non-monetary prize		
Final vs. provisional testing		
Provisional testing type		
Task type		
Submission type		
Machine learning		
Task goal		
Task goal is to improve exist. solution		

Table 4: Description of the matched challenges ($N = 52$).

Variable name	Category	Mean / Freq.	St. Dev.	Min	Max
Scoring type	Absolute	40.38% (21)			
	Relative	59.62% (31)			
Number of contestants		122.52	80.33	10	293
Contestants/registrants ratio		0.77	0.31	0.14	1.00
Duration		289.63	96.09	8	355
Average contestant experience		29.50	50.30	1.16	221.53
Engagement barrier		1.89	0.29	1	2
Scoring complexity		1.32	0.44	1	2
Prize pool		0	0	0	0
Number of prizes		0	0	0	0
Non-monetary prize	No	100%			
Final vs. provisional testing	Similar	100%			
Provisional testing type	Shared	92.31% (48)			
	TC only	7.69% (4)			
Task type	Optimization	67.31% (35)			
	Strategy	32.69% (17)			
Submission type	Code	100%			
Machine learning	No	100%			
Task goal	Quality	96.15% (50)			
	Resource or both	3.85% (2)			
Task goal is improve exist. sol.	No	100%			

Table 5: Description of the contestants in the matched challenges.

Variable name	All (N = 5,916)				In both scoring types (N = 1,239)			
	Mean	St. Dev.	Min	Max	Mean	St. Dev.	Min	Max
Number of unique individuals	2774				319			
Rating[1]	0.506	0.288	0.004	1	0.565	0.290	0.004	1
Number of prev. challenges	8.204	13.175	0	125	19.718	18.345	1	125
Code complexity[2]	117.180	115.361	1	1476	147.124	128.391	1	1208
Proximity to rating[1]	0.813	0.162	0.062	1	0.830	0.151	0.181	1
Total code lines written[3]	783.165	1120.635	3	37076	1006.899	1123.781	3	20701
Number of submissions	7.864	9.364	1	162	10.423	11.312	1	147

Note:

1. Based on N = 4,158 observations for all.
2. Based on N = 5,577 observations for all and N = 1,195 for individuals in both scoring types.
3. Based on N = 5,113 observations for all and N = 1,123 for individuals in both scoring types.

Table 6: Results for the timing, length, and complexity of the first submission.

	Timing		**Length**		**Code complexity**	
	(C1.1)	(C1.2)	(C2.1)	(C2.2)	(C3.1)	(C3.2)
Relative scoring		**0.319**		**−0.353***		**−0.441***
		(0.174)		**(0.175)**		**(0.217)**
Relative scoring × Rating		**−0.400**		**1.442***		**1.274***
		(0.218)		**(0.577)**		**(0.615)**
Relative scoring × Rating^2				**−1.218***		**−0.998**
				(0.530)		**(0.562)**
Rating	0.183	0.411	2.685**	1.848	3.043**	2.361*
	(0.596)	(0.613)	(0.967)	(1.027)	(0.997)	(1.062)
Rating^2	−0.368	−0.368	−4.659*	−3.891	−5.510*	−5.003*
	(0.548)	(0.550)	(2.127)	(2.156)	(2.189)	(2.222)
Rating^3			3.027*	2.977*	3.608**	3.639**
			(1.359)	(1.357)	(1.396)	(1.395)
Number of prev. challenges	−0.004	−0.004	−0.002	−0.002	−0.002	−0.002
	(0.003)	(0.003)	(0.002)	(0.002)	(0.002)	(0.002)
Constant	−0.706***	−0.893***	4.759***	4.956***	3.396***	3.639***
	0.180	(0.206)	(0.166)	(0.194)	(0.182)	(0.222)
Observations	1229		1029		1188	
Individuals	317		280		308	
Challenges	42		40		42	
Strata	12		11		12	
Individual-level variance	0.374	0.374	0.196	0.194	0.192	0.190
Challenge-level variance	0.053	0.053	0.076	0.078	0.187	0.187
Stratum-level variance	0.064	0.064	0.087	0.084	0.110	0.111
Log Likelihood	534	534	−6395	−6391	−5848	−5845
AIC	−1048	−1048	12807	12807	11714	11714
$\Delta\chi^2$	**3.869 (2 df)**		**6.630 (3 df)**		**6.847 (3 df)**	

Note: Coefficients and standard errors (in brackets) for: (C1) mixed effects beta regression for the timing of the first submission, measured as proportion of the challenge duration; (C2) mixed effects gamma regression for the number of lines of the first submission; (C3) mixed effects gamma regression for the cyclomatic complexity of the code in the first submission. The models include weights by stratum and control for treatment ordering effects using the number of previous challenges the coder has participated in. The models also include random intercepts by individual, challenge, and stratum.

* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$

Table 7: Results for performance, likelihood to return, and effort for first-time contestants.

	Code complexity	**If returns**	**Total code lines**	**Num. submiss.**	**Time in chall.**
	(C4)	(C5)	(C6)	(C7)	(C8)
Relative scoring	**0.052**	**−0.256**	**0.050**	**−0.147**	**−0.231***
	(0.148)	**(0.216)**	**(0.158)**	**(0.129)**	**(0.114)**
Constant	4.416***	0.221	6.204***	1.898***	−0.985***
	(0.144)	(0.223)	(0.141)	(0.105)	(0.107)
Individuals	1159	1198	1055	1198	1198
Challenges	41	41	41	41	41
Strata	13	13	13	13	13
Challenge-level var.	0.142	0.200	0.146	0.093	0.046
Stratum-level var.	0.102	0.268	0.066	0.017	0.037
Log Likelihood	−6178	−726	−7658	−3436	1682
AIC	12366	1461	15326	6882	−3355

Note: Coefficients and standard errors (in brackets) for: (C4) mixed effects gamma regression for the cyclomatic complexity of the code in the final submission; (C5) mixed effects logistic regression for the likelihood to participate in at least one other challenge on Topcoder; (C6) mixed effects gamma regression for the the total number of code lines the coder wrote in the challenge; (C7) mixed effects gamma regression for the the number of submissions the coder made in the challenge; (C8) mixed effects beta regression for the duration of the coder's participation in the challenge, measured as the difference between the time of their last and first submissions, divided by the duration of the challenge. The models include weights by stratum and random intercepts by challenge and stratum.
* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$