To Share, or Not to Share? Community-Level Collaboration in Open Innovation Contests

YLA TAUSCZIK, College of Information Studies, University of Maryland
PING WANG, College of Information Studies, University of Maryland

Open innovation contests have been incredibly successful at producing creative designs and solutions. While participants compete for prizes in these contests, over half of contests conducted on online platforms allow participants to share ideas during contests, for benefits such as individual learning, community building, and cumulative innovation. Such sharing, however, is at tension with the competitive nature of crowd contests. To understand this tension, this study investigates community-level sharing of code on Kaggle, a contest platform for predictive modeling. Analyzing data on 25 contests in 2015 and 2016, we find that 10% of users shared code during contests, that participants doing medium well in the contest were the most likely to share code, and that sharing code improved individual, but not collective performance. These findings allow us to contribute insights about the participants, conditions, processes, and outcomes of community-level collaboration to both research on and design of open innovation contests.

Additional Key Words and Phrases: Crowdsourcing; Innovation contests; Open collaboration; Open innovation; Kaggle

ACM Reference format:
Yla Tausczik and Ping Wang. 2017. To Share, or Not to Share? Community-Level Collaboration in Open Innovation Contests. *Proc. ACM Hum.-Comput. Interact.* 1, 2, Article 100 (November 2017), 23 pages.
https://doi.org/10.1145/3134735

1 INTRODUCTION

Contests have become a successful way organizations engage crowds to create innovative solutions to organizational problems. Early contests, such as the Netflix Prize, were organized and hosted by individual organizations. Now, many take place on specialized platforms which have emerged to host contests in specific areas, such as InnoCentive (R&D), TopCoder (programming), Dell's IdeaStorm (idea generation), and Kaggle (analytics and predictive modeling). Contests on these platforms are sponsored by a mix of individual companies, public organizations, and nonprofit organizations [9]. These contest platforms can provide organizations with easier access to knowledge and expertise outside their organizational boundaries, taking advantage of open innovation [10, 22].

This work was supported by the National Science Foundation (CISE IIS-1546404).

2573-0142/2017/11-ART100 $15.00
https://doi.org/10.1145/3134735

From an organizational perspective, open innovation contests lower the costs associated with maintaining internal expertise, both minimizing the risks associated with R&D and broadening the pool of available knowledge and solutions. From an individual perspective, open innovation contests offer monetary prizes, professional recognition, learning opportunities, and challenging problems.

The Netflix Prize was one early and influential innovation contest. In October 2006 Netflix released a large data set of customer reviews for movies and challenged the data science community to beat its in-house recommendation system by at least 10%. The winner with the best performing algorithm above this threshold was promised a million dollar prize. By June 2007, two thousand teams submitted solutions; many performed better than the internal system, though none reached the contest's 10% improvement goal [5]. Three years later, two teams finally met the target; the competition was declared a success; and the prize was awarded to one of the winning teams [4].

Research on open innovation contests has primarily focused on modeling individual competitive behavior and the design elements needed to make contests popular and successful [2, 3, 12, 28]. However, the nature of open innovation contests–strict time limits, complex tasks, and an emphasis on innovation–makes teamwork and collaboration advantageous [12]. In fact, though open innovation contests tend to emphasize the competitive aspects of contests, more than half provide avenues for collaboration through discussion and sharing ideas among contestants [9]. In the case of the Netflix Prize, competing teams shared code, data, and insights about the data, explained their algorithmic approaches, and combined their algorithms [5]. Early notable breakthroughs in algorithms were adopted by later teams and then later teams developed hybrid algorithms that combined multiple approaches. One of the teams tied for first place three years later, "The Ensemble", was an aggregation of 23 teams from earlier rounds who had merged their algorithms and their work [4].

Hybrid contest platforms that establish a competitive environment among participants while also providing opportunities for collaboration through sharing ideas, partial results, and feedback may be ideal [8]. Open innovation contests have been incredibly successful at producing creative designs and solutions to difficult problems, because individuals from a large crowd compete to identify the optimal solution (e.g. [20]). From foundational work on open collaboration and online communities, we know that open, large-scale collaborations can create impressive artifacts and innovations [14, 27]. However, on hybrid contest platforms community-level collaboration is at tension with the competitive nature of the crowd contests. While the competitive aspects of these platforms have been studied in detail [1], we do not know as much about how collaboration unfolds on these sites.

The contribution of this paper is to understand whether collaboration can thrive in innovation contests, which are highly competitive, zero-sum games. We investigate community-level collaboration in 25 contests on one site, Kaggle–an online innovation contest platform for analytics and predictive modeling. On Kaggle, participants compete for monetary prizes, reputation points, and sometimes job opportunities. Kaggle also allows participants to share code during contests. We ask four exploratory questions to help understand collaboration: who shares code; how the design of a contest affects code sharing; whether code sharing impacts individual and collective performance; and how code evolves over time during these contests. These questions address three major objectives: 1) whether the nature of competition and the specifics of contest design inhibit individual participation in collaboration, 2) whether collaboration offers individual and/or collective benefits as would be expected from the open innovation literature and 3) whether collaboration follows patterns expected from studies of cummulative innovation. To date the vast majority of studies of innovation

contests have focused on competition and performance and have ignored community-level collaboration. There are a few exceptions which offer preliminary findings each focusing on a single contest. [9, 15, 19] are primarily qualitative and exploratory, while [8] is experimental and quantitative, but studied collaboration in a contest in which participants were not given the option to keep work secret. The current study complements and extends these studies by taking a quantitative approach to studying collaboration and studying many more contests. Understanding collaboration in innovation contests is important because innovation contests may provide better experiences to participants and better outcomes to host organizations when contests cultivate both collaboration and competition rather than purely supporting competition.

2 RELATED WORK

Innovation contests are a historically important R&D strategy that have elicited innovations ranging from incremental improvements to radical breakthroughs [1]. To date, over two hundred academic publications have conducted research on innovation contests, a majority focusing on online innovation contests using the crowd [1]. We summarize the relevant literature on innovation contests and related crowdsourcing initiatives as it pertains to collaboration, motivation, and contest design.

2.1 Collaboration in Innovation Contests

There are two types of collaborations in innovation contests: within-team collaboration and community-wide collaboration. Teams were found to outperform individuals in contests on Kaggle [33], presumably because team members were able to collaborate within their team. Dissanayake and colleagues [12] found that teams on Kaggle composed of skilled members and with leaders socially connected to the community did better than other teams.

A few papers have directly investigated community-level collaboration on innovation contest platforms. They found that some individuals chose to collaborate and compete while others only competed. Researchers analyzing comments from an online German design contest [9] identified two distinct sets of users, ones which did not collaborate and another set which did. They further conducted focus groups among participants, identifying several reasons for collaboration including curiosity, a desire to recognize good work done by others, and a desire to help individuals who were not doing well. More strategically, participants reported collaborating to learn from others, to reconsider their own work in light of others' ideas, and to integrate ideas from other contestants. Participants who did not collaborate reported having a clear idea of what they were trying to accomplish. They ignored external discussion to focus on their own idea. Similarly, Hutter and colleagues [19] analyzed the content and network structure of comments among contestants in an online design competition, finding that contestants engaged in a variety of strategies, some exclusively competitive, some exclusively cooperative, and some a combination of the two. In a jewelry design contest, participants provided each other with thousands of peer-evaluations and comments, presumably creating a sense of community [15]. A follow-up survey revealed that those users who felt the strongest sense of community also reported having the best experience.

Community-level collaboration was found to be associated with better collective performance in one experiment. Boudreau and Lakhani [8] conducted a field experiment on TopCoder, a crowd programming contest platform. They experimentally compared outcomes for two contest designs, a purely competitive design in which no code sharing was allowed and an open design in which all submitted code was made publicly available during the

contest. The crowd in the open design condition produced better performing code, although fewer people participated. The crowd was able to do better with an open design because they learned from each other, with the most successful ideas spreading throughout the crowd. In addition, they were able to build on each others' work in a process known as cumulative innovation.

These few papers suggest collaboration is beneficial in innovation contests, whether it occurs within a team or at a community level, however only some individuals choose to collaborate. More research is needed to understand who collaborates, under what conditions, and how collaboration unfolds. In addition, collectively this limited set of papers has only studied community-level collaboration in a few innovation contests.

2.2 Open Collaboration

Related work from online communities and crowdsourcing also addresses the value of large-scale collaboration and the trade-offs associated with combining collaboration and competition. Literature on open collaboration suggests that one of the main advantages of online communities is their ability to create knowledge collaboratively [13]. In these communities, participants openly reveal their knowledge and expertise and work together [27]. Through collaboration ideas are refined, integrated, and recombined in complex ways fueling innovation. Combining and recombining design ideas generated by the crowd increases the creativity of the designs [30]. Large-scale collaboration may be especially important in promoting creativity through cumulative innovation.

Other types of crowdsourcing initiatives such as the Stack Exchange Q&A platform (e.g. Stack Overflow) also combine competition and collaboration. Researchers find that collaboration happens at a low but meaningful level on these platforms and improves performance where it occurs [21, 25]. Collaboration may not be as prevalent as it could be on these platforms because of the competitive nature of these Q&A communities. When the researchers compared similar types of posts on Stack Overflow and a discussion mailing list, they found more collaboration on the mailing list [31]. They argue that Stack Overflow is less collaborative than mailing lists because it uses a reputation system, in which individuals can compete to win points, incentivizing non-collaborative behaviors.

This stream of research suggests that collaboration is beneficial, but that competition may reduce the frequency of substantive, valuable collaboration. This research has been conducted on other types of online platforms with different community structures, but may be applicable to innovation contests.

2.3 Motivation

Individuals who participate in crowdsourcing initiatives and, in particular, innovation contests are motivated by both intrinsic and extrinsic incentives. Zheng and colleagues [32] found that both intrinsic incentives, such as enjoyment, curiosity and personal challenge, and extrinsic incentives, such as earning money and gaining reputation, were important in motivating participation on Taskcn.com, a Chinese crowdsourcing platform. Similarly, Yu and colleagues [29] found that intrinsic motivations in the form of social motivations and learning were just as useful for motivating crowd work as the monetary compensation offered in an experiment run on Amazon's Mechanical Turk. Boons and colleagues [6] found that being proud to belong to the crowdsourcing community, a social and intrinsic motivation, was related to greater engagement on an innovation contest platform. In fact, intrinsic incentives may be slightly more important than extrinsic incentives on these platforms [32].

Although, researchers find evidence that both intrinsic and extrinsic motivation encourages participation in crowd contests, previous research in other contexts suggest that intrinsic and extrinsic motivations can sometimes be at odds with one another. Deci and colleagues [11] found that adding extrinsic incentives can reduce intrinsic motivation previously associated with a task. In several experiments combining intrinsic and extrinsic incentives on a crowdsourcing platform, researchers found that some types of extrinsic incentives undercut intrinsic incentives while others did not [29]. The researchers argued that extrinsic and intrinsic incentives can be combined synergistically as long as they are carefully designed to have non-conflicting goals. In line with this argument, Rogstadius and colleagues [24] found that they were able to combine intrinsic and extrinsic motivators to increase the quality of crowd work.

The benefits of community-level collaboration are primarily intrinsic, such as learning, helping others, and feeling part of a community, while the benefits of competition are primarily extrinsic, such as building a reputation and winning prizes. This stream of research suggests that depending on the specific nature of the incentives extrinsic incentives may or may not weaken intrinsic incentives. Contest design as well as an individual's position within the community may affect the nature of these incentives, thus affecting whether extrinsic incentives weaken intrinsic incentives, and ultimately affecting whether or not individuals are motivated to collaborate.

2.4 Contest Design

In 2009, Bullinger and colleagues [9] investigated 69 open innovation contests and identified common design differences between contests. Major differences included different types of sponsoring organizations, different contest durations, different types of rewards offered to participants, and whether participants competed as individuals or as teams. A few researchers have investigated how some differences in design contests affect participation and performance. Yang and colleagues [28] found that specific contest design elements did affect the number of participants and the proportion who submitted final work. They found that contests obtained more submissions when they had longer durations and were not as complex. When contests had higher monetary rewards more people entered the contest, but fewer submitted final work.

This stream of research suggests that contest design has a significant impact on participants' behavior during contests. However, these studies have only examined how contest design affects the rate at which individuals participate (e.g. make submissions) and how they perform. To date no research has examined how contest design affects whether or not individuals collaborate during a contest.

2.5 Summary

In summary, community-level collaboration has not been studied extensively in innovation contests. First, the few studies that have examined this type of collaboration in innovation contests suggest that collaboration is beneficial but these studies did not investigate the participants, conditions, and processes of such community-level collaboration. Second, work on other types of online platforms suggests that community-level collaboration may be beneficial but may be dampened by competition, a pattern that has not yet been examined in innovation contests. Third, research on motivation shows that intrinsic motivations associated with collaboration and extrinsic motivations associated with competition can sometimes be at odds on open innovation platforms, but this research stream is inconclusive as to whether competition weakens collaboration. Lastly, research on contest design reveals that specific

configurations of intrinsic and extrinsic incentives influence participation and performance, but has not yet examined the effects of contest design on collaboration at the community level.

The current study examines community-level collaboration in open innovation contests, addressing this knowledge gap while extending and complementing all four streams of research mentioned above. Foremost, this study adds to the few studies on collaboration in innovation contests by examining the participants, conditions, and processes of collaboration. Also, this study's framing of user behavior in terms of a competitive-collaborative tension in innovation contests enriches extant research on collaboration in open innovation platforms. Further, this study demonstrates which contest design features affect collaboration despite the competitive nature of innovation contests, contributing new insights to research both on motivation and on contest design.

3 CURRENT STUDY

We investigated community-level collaboration on Kaggle, an online, open innovation contest platform. On Kaggle, participants compete with each other in analytics and predictive modeling contests, either as individuals or in small teams. In the spring of 2015, Kaggle introduced a new feature which allowed individuals to openly share code with the community as they worked on a contest. Shared code can be executed, modified, and submitted as a solution to a contest. These features of platform and contest design allow us to investigate the tension between competition and collaboration in open innovation contests. In this study, we specifically analyzed code sharing in 25 contests from 2015 to 2016 and addressed four exploratory research questions to help understand community-level collaboration.

3.1 Research Question 1: Who shares code?

Unlike collaboration within teams on Kaggle, in which cooperation and competition work in concert, community-level collaboration through code sharing pits cooperation against competition. When participants openly share code, they are sacrificing the advantages of keeping their work secret. When sharing and secrecy point to different advantages, a tension develops between cooperation and competition, and between sharing and secrecy [26]. By sharing code individuals may benefit from getting feedback, having others improve their code, learning through co-creation, helping others, and feeling part of a community [9, 15]. By keeping code secret individuals may benefit from performing better relative to others which may lead to higher rankings on leaderboards, winning prize money, and job opportunities.

Individual differences are likely to affect the trade-off between sharing and secrecy. We expected that the advantages of secrecy would be greatest for individuals who were performing better on the site and within a contest. More skilled contestants prefer competition over cooperation [7]; skilled contestants have the most to gain from secrecy because they have a higher chance of winning prize money and being recognized on leaderboards. We also expected that individuals who had fewer opportunities to collaborate within a team, because they were working alone or in a smaller team, would be more likely to share code because they would have more to gain from collaborating with the community at large. We evaluated the effects of individual performance (site-wide and contest-specific) and team membership on the likelihood of an individual to share code.

3.2 Research Question 2: How does the design of contests affect code sharing?

Open innovation contests vary in their designs, such as whether they allow teams to compete [9]. We investigated whether four design parameters–the size of the monetary reward, whether

team size was restricted, the complexity of the problem, and the length of the competition–affected the amount of code sharing. Situational conditions can affect the trade-off between sharing and secrecy [26]. We predicted that as individual short-term extrinsic incentives increased (e.g. size of the reward) there would be less code sharing because there would be a greater advantage of secrecy. On the other hand, as it becomes more difficult to compete individually or as a small team (e.g. more complex problem, restricted team size, shorter contest time) there would be more code sharing because of the greater advantages of sharing and collaborating.

3.3 Research Question 3: Does code sharing affect individual and collective performance in contests?

Previous research suggests that community-level collaboration is individually and collectively advantageous. Many of the top contestants in two open innovation contests studied had participated in community-level discussions of design ideas [9, 19]. Individuals can benefit from sharing code in the short term, by gaining valuable feedback, borrowing ideas from others, and combining others's ideas with their own, and in the long term by learning from others's expertise and from the co-creation process that happens when individuals work together to develop code. Widespread sharing of code can enable cumulative innovation, in which individuals improve, combine, integrate and build on each others work collaboratively. Code sharing has been associated with improved collective performance in an online programming contest; there was improvement in both the average and top solutions [8]. We investigated how sharing code affected individual and collective performance.

3.4 Research Question 4: What code gets modified?

One of the most important potential benefits of community-level collaboration through sharing code is that it might enable cumulative innovation, in which code is improved, combined, and built upon by the community at large. One way researchers have studied cumulative innovation is by examining changes in innovation networks over time, such as growth of patent citation networks. We applied findings from one such study by Podolny and Stuart [23] to see if we saw similar patterns on Kaggle. We translated the findings for patent citations networks to code branch networks to understand what code was modified on Kaggle.

In examining patents, Podolny and Stuart [23] found that areas of innovation that were seen as more legitimate attracted more attention, driving more innovation in these areas. They also found that individuals used patent author reputation as a measure of legitimacy. One way users may evaluate the legitimacy of code on Kaggle is by examining who wrote the code or by examining how other community members rated the code. Thus, we predicted that code that was written by a highly-ranked participant and/or given a higher community rating would be seen as more legitimate and would attract more modification.

Podolny and Stuart [23] also found that some areas of innovation attracted more innovation simply because more individuals came across them. If individuals only explore a limited search space starting from where they are familiar, they may only uncover and be inspired by nearby innovations. On Kaggle we suspected that code that was easier to understand because it used familiar libraries, functions, and approaches might be more accessible than code that was more distinctive and unique. We predicted that code that was more prototypical, in that it was more similar to other code that had been shared previously, would attract more modification. We made this prediction because we believed that prototypical code would be explored first by more people because it was more familiar.

Finally, Podolny and Stuart [23] found that areas of innovation that were already heavily exploited were less likely to grow. Areas that are heavily exploited have less potential for new innovation. We expected to find a conceptually analogous pattern on Kaggle. We suspected that code that had only produced minor modifications had less potential for further exploitation than code that had produced a diverse set of modifications. Thus, we predicted that code that had only produced minor modifications would be less likely to attract further modification. This fourth research question is important for two reasons. First, by tracking how code evolves it describes how collaboration unfolds. Second, by comparing how code evolves on Kaggle to how patent citation networks grow it tests whether collaboration on Kaggle is similar to collaboration in other cumulative innovation settings.

4 METHOD

4.1 Description of Site

Kaggle describes itself as a platform for data scientists and data enthusiasts. The platform organizes programming contests, in which the crowd competes, individually or in teams, to solve predictive modeling and analytics problems. However, Kaggle also supports a variety of other avenues for data scientists to discuss and collaborate on data analytics, including hosting discussions on open data sets, code repositories, discussion forums, and practice problems. The platform serves several different functions: providing organizations with access to data scientists and the solutions they create, providing real-world problems and data sets to data scientists working to advance machine learning, providing a learning environment in which users can become more skilled at machine learning, and allowing companies to recruit from a pool of talented data scientists.

Kaggle regularly hosts contests sponsored by external organizations such as Google, Intel, and the Nature Conservancy. For each contest the crowd is provided a description of the problem, a public data set, and a set of evaluation criteria. Contests are time-limited, running anywhere from a few weeks to a few months. Contests award prizes to the participants with the best performing solutions. In our data from 25 contests all prizes were monetary. Different contests have different rules (e.g. some restrict team size). Contestants typically submit multiple solutions over the course of the contest, although they may be limited to one submission per day. Contestants are ranked based on the performance of their best submission. Rankings are displayed publicly on a leaderboard during contests. Individuals earn points for their performance in contests. Kaggle allots points[1] according to the following formula, which penalizes individuals who compete on teams with more teammates (individuals competing alone are considered to have a team size of 1) and rewards individuals with higher rankings when there are more total teams competing (individuals competing alone are each counted as their own team):

$$\frac{100000}{\sqrt{N_{teamsize}}} \times Rank^{-0.75} \times log_{10}(1 + log_{10}(N_{teams}))$$

Points from contests accumulate over time and are used to rank individuals on the site. For site-wide rankings, Kaggle applies a decay function ($e^{\frac{-t}{500}}$) so that points in more recent contests count for more.

In April 2015 Kaggle released a new feature called kernels (originally called scripts). This feature allowed users to run code directly on Kaggle and to share code publicly with others. Kaggle allowed code in a few commonly used languages: R, Python, Julia, and SQLite. Users

[1] https://www.kaggle.com/progression

could browse code related to a contest; and rate, modify, and comment on the code (Figure 1).

4.2 Code Sharing

Users tend to share three main types of code: solutions to the prediction problem, code that helps individuals understand the data better, and helper code that makes other code easier to implement. For example, in response to the West Nile Virus contest, in which users were tasked with predicting when and where in the city of Chicago mosquitoes would test positive for West Nile Virus, one user shared code that applied a random forest algorithm to solve the prediction problem ("Beating the Benchmark ;) (0.71+)") while another user shared code that applied logistic regression ("Starter Logistic Regression in R"). This type of code can be directly submitted to the contest or modified and then submitted. Users also share code that helps them understand the data better. Understanding the data is important for developing a good solution and fine tuning the performance of predictive models. For example, one user developed and shared code that visualized the locations in Chicago where mosquitoes tested positive for West Nile Virus ("West Nile heat map"), while another user shared code that outputted descriptive statistics for the most relevant variables ("Exploring features"). Finally, helper code makes other code easier to write or more efficient, a user shared code that converted the data from a dense to a sparse representation reducing the amount of memory that was needed to run other code on the data ("Using 10x less RAM").

Modifications to shared code are often minor tweaks, but also involve incremental improvements to boost performance. Common ways that individuals improve on solutions is to add feature selection ("RandomForestClassifier Minus a Few", "XGBoost after feature pruning") or to tune parameters ("XGBoost parameter tuning with GridCV"). These modifications do not change the main algorithms underlying the predictive model, but can boost performance, and often make a difference in terms of scores given to final solutions.

4.3 Description of Data

Kaggle made a subset of its data publicly available as a downloadable database[2] on July 20, 2016. Almost all contests up to July 2016 are included in this data set with a few exceptions. Contests that focused on recruitment were not present (N = 16, with one exception). Contests that were limited to only certain types of participants, such as students in an online course, higher ranked users only (masters level), or invitation only, were not present in this data set (N = 8).

4.3.1 Inclusion Criteria. Kaggle only gives participants points for performance in some types of contests. In particular, they exclude contests designated as "getting started", "playgroup", or "in class". We applied a similar criteria in analyzing the data. We excluded these contests from point calculations and analyses, which left 150 contests that were considered in point calculations. All our analyses focused on contests that supported kernels for the majority of each contest. Among the 150 remaining contests, 25 contests from 2015 to 2016 occurred after Kaggle introduced code sharing as a feature and thus these contests were included in our subsequent analyses.

4.3.2 Metrics and Variables. All of our analyses focused on how much novel and modified code participants shared in a contest. We focused on sharing rather than using shared code, because the data provided by Kaggle did not provide good enough records of how

[2]https://www.kaggle.com/kaggle/meta-kaggle

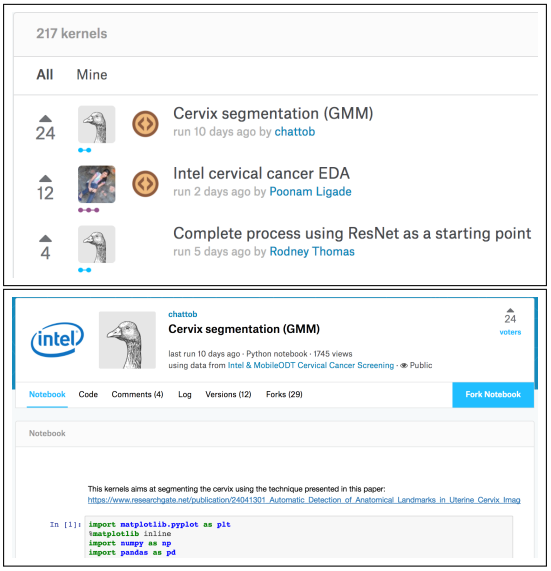

Fig. 1. Kaggle provides a list of shared kernels visible to the community for each contest; an example of part of a list is featured in the upper panel of the figure. Participants can display, execute, and fork shared kernels; a screenshot in the lower panel shows how code is displayed to participants.

code was used. For many of our analyses we examined how behavior in the first half of a contest affected behavior in the second half of the contest[3]. Within the limitations of an observational study this allowed us to partially control for the directionality of effects. We chose to divide contests in half based on duration because duration is invariant to user behavior; it provides the largest time intervals to observe early and late behavior; and it is a typical method used in studying changes over time in work groups (e.g. [16]).

Participant Variables. We limited analyses to users who had submitted in at least one contest. For each participant we calculated the cumulative number of **site points** they had earned on the site at the beginning of each contest. We used Kaggle's formula to calculate points per contest (see Description of Site), but chose to aggregate across contests without applying a decay function. Instead, we controlled for how long users had been active on the site by recording the total **number of contests** a user had competed in at the time of the contest and including it as a covariate in our models. For each contest, we also recorded the size of the team a user was competing with (**team size**); the number of points earned in the first and second half of the contest (**contest points**); and whether they **shared code** in the first and/or second half of the contest (see Kernel Variables). Where relevant we controlled for whether they **submitted code** in the first half of the contest.

Contest Variables. Contests differed in their designs and user behavior. In terms of their design we recorded whether a contest restricted the size of teams (**restricted teams**), the total amount of money offered to prize winners (**reward quantity**), the complexity of the problem as operationalized as the number of bytes of data that had to be analyzed

[3]The first and second half of the contest were defined to divide the duration from the start to the end of the contest equally. Typically more activity was present in the second half of a contest, but both were active periods.

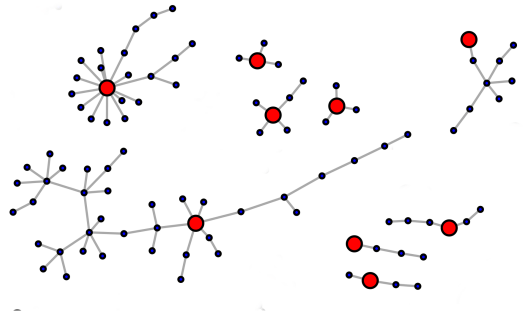

Fig. 2. Examples of kernels that were shared and then expanded into branches by others forking the novel code. Large red nodes represent novel code, edges connect novel code to forked code, forked code is represented by small black nodes.

(**data size**), and the number of days given to submit a solution (**contest length**). For each contest we also recorded the **percent of users who shared code** and the total number of modifications made to shared code (**number of forks**) as measures of community-level collaboration. We measured collective performance in two ways, by calculating the percentage improvement in the the maximum and the median scores from a contest's midpoint to its end (**% improvement in top and median solutions**). Where relevant we controlled for the total **number of participants** who submitted solutions to a contest.

Kernel Variables. Users shared code by creating kernels. For each kernel we have a record of the time when the code was shared, the code itself, the user who shared it, its revision history, and a list of votes. We distinguished between three types of shared code: novel code (bases), modifications of novel code written by others (forks), and non-modified copies of novel code (omitted from calculations and analyses). For the fourth research question we only examined bases that had been created in the first half of a contest. For these bases, we recorded the **kernel author's site points** at the start of the contest, the **kernel author's contest points** at the midpoint of the contest, and the total number of votes a kernel had received up until the midpoint of the contest[4](**community rating**).

For the fourth research question we divided code into branch families, that is a base and all of its forks (Figure 2). For each base created in the first half of a contest we measured whether a base had received any forks in the second half of the contest (**kernel forked**). For some models we controlled for the total number of forks created in the first half of contest (**branch size**).

Based on methods from natural language processing, we calculated the average distance between a base in the contest and all other bases in *previous* contests[5] (**code base distance**), providing a measure of how (dis)similar a kernel was to the established code base. We also calculated the average distance between a base and its forks created in the first half of the contest (**branch distance**). To calculate these distances, we treat the code of each kernel as a document. We then tokenized the entire corpus of kernels into words weighted

[4]To remove interdependencies in the data we excluded votes given by users who went on to modify the kernel. Thus community rating only included the rating by community members who did not modify the base kernel.

[5]We limited comparison to previous bases to bases in our set of 25 contests as these were the primary contests that allowed kernels and therefore had shared bases.

	Mean	Median	Range
Participants			
% Share Code	10.0%	-	-
Team Size	1.34	1	1-18
Num. Contests	2.59	1	1 - 104
Site Points	2,830	483	52 - $3.0 \cdot 10^5$
Contest Points	819	372	52.0 - $6.7 \cdot 10^4$
Contests			
Restricted Teams	20.0%	-	-
Reward Quantity	$32K	$25K	$250 - $100K
Data Size	7,962MB	101MB	1 - 88,289MB
Contest Length	72 days	67 days	39 - 160 days
Num. Participants	1,589	1,214	407 - 5,696
Num. Forks	233	223	0 - 758
% Improvement			
Top Solution	12.8%	0.3%	0 - 270%
Median Solution	22.1%	3.2%	0 - 214%
Kernels			
Kernel Forked	27.5%	-	-
Community Rating	1.70	0	0-157
Branch Size	2.05	1	1-65

Table 1. Descriptive statistics.

by term frequency-inverse document frequency (tf-idf). Using this set of dimensions we then computed cosine similarity between pairs of kernels. We took the inverse as a measure of distance between two kernels.

5 RESULTS

5.1 Research Question 1: Who shares code?

Contest participants face a trade-off between cooperation and competition, in which they can choose to cooperate without competing, compete without cooperating, or do a mix of both. We predicted that individual differences would influence the trade-off and lead some individuals to be more likely to share code. Overall 10% of participants shared code during a contest (see Table 1). We performed two mixed-effects logistic regression models to predict which contest participants were more likely to share code during the first and second halves of the contest (see Table 2). The contest ID was included as a random effect, to control for within-contest differences. Some participants had been active on the site longer and had more opportunities to accumulate site points, we included the number of contests as covariate to control for experience on the site. For the second model we also included activity in the first half as control variables. We considered quadratic effects for individual performance measures.

We predicted that participants who were performing better on the site and in the contest would be more secretive and thus less likely to share code. Contrary to our prediction individuals who had accumulated more points on the site at the start of the contest were more likely to share code. There was a positive relationship between site points and code

	Model 1 Shared Code 1^{st} Half		Model 2 Shared Code 2^{nd} Half	
	Coef.	SE	Coef.	SE
Intercept	-3.44	0.16	-2.79	0.12
Team Size	-0.039	0.02	-0.068*	0.029
Site Points	0.064***	0.0066	0.041***	0.0067
Contest Points	-	-	2.11***	0.42
Contest Points^2	-	-	-0.14***	0.030
Num Contests	0.015***	0.0024	0.0022	0.0029
Shared Code	-	-	1.92***	0.065
Submitted Code	-	-	-8.65***	1.46

Table 2. Mixed effects logistic regression models with 39,725 observations and 25 groups. Model 1 and Model 2 display the effect of participant performance and team size on sharing code in the first and second half of the contest respectively. Predictors used in Model 2 (contest points, shared code, submitted code) were calculated based on the first half of the contest only. * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

sharing in the first and second halves of the contest[6]. Partially in support and partially in contradiction with our prediction, individuals who were doing well in the contest, but not at the very top, were the most likely to share code (Figure 3). This is evidenced by a quadratic relationship between contest points in the first half and sharing code in the second half of the contest. One explanation of these results is that individuals who are doing better on the site and within a contest are more motivated to share code.

However, individuals who are doing the very best in a contest hold back code. The top $\frac{1}{2}\%$ of individuals in a contest shared code at rates similar to individuals performing in the bottom half of the contest (i.e. 50 percentile and below) and at half the rate of that of those in the 80th-90th percentile ($\sim 4\%$ vs. 8%). These individuals have the most to lose from sharing the code that has put them ahead of everyone else. These results suggest skilled individuals make context-sensitive choices about whether or not to collaborate at a community-level based on their performance in a contest, rather than displaying a static predisposition toward competition over cooperation.

We also predicted that some participants, such as those who had fewer opportunities to collaborate within a team, would be more likely to share code as a way to gain the benefits of collaboration. Though team size was unrelated to sharing code during the first half of a contest, we did find a negative relationship with code sharing code during the second half. This (partially) supports our prediction, suggesting that individuals who work alone or in smaller teams are more likely to share code in the second half of the contest. When individuals cannot collaborate as much within a team, they may resort to collaborating with the community at large.

5.2 Research Question 2: How does the design of contests affect code sharing?

Contests on Kaggle vary in their rules. For example, they offered different amounts of prize money, ranging from $250 to $100,000; they varied in duration from one month to nearly 6 months; they required the analysis of data of very different sizes; and 20% restricted team

[6]The quadratic effect of site points was not significant so it was dropped from the model to improve fit.

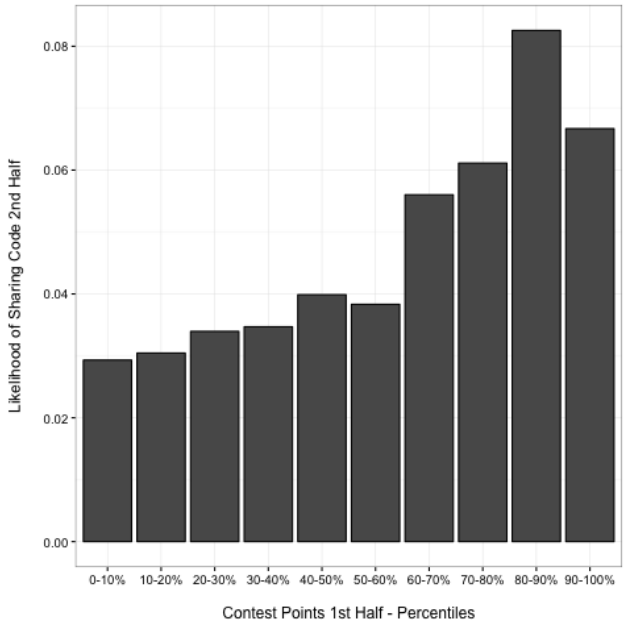

Fig. 3. The relationship between contest participants's points in the first half of a contest and their likelihood of sharing code in the second half of the same contest.

size (Table 1). We predicted that differences in contest design would shift the trade-off between cooperation and competition, and between sharing and secrecy, such that some designs would be associated with more code sharing. Using the sample of 25 contests, we built two linear regression models, one predicting what percentage of users shared code during a contest and the other predicting the magnitude of collaboration as indicated by the amount of modified code (Table 3). The contest design characteristics did not heavily covary with each other, correlations among characteristics ranged from non-significant to small ($r = 0$ to 0.37).

We predicted that as individual incentives to win (i.e., monetary rewards) increased, there would be less code sharing because there would be a greater opportunity cost to sharing. We found no relationship between the prize amounts and the percentage of users who shared code or the degree of collaboration. Since so few individuals win prizes, and the people sharing code are less likely to be the prize winners, the prize amount does not seem to affect sharing behavior.

We also expected that as it became more difficult to compete individually or as a small team there would be more code sharing because there would be more to gain from collaborating as a community. We found that shorter contests were associated with a greater percentage of users sharing code and that there was more total collaboration for contests that restricted team size. The size of the data to be analyzed was not associated with differences in sharing or collaboration. Given more time pressure, individuals may see more benefit to collaborating by getting feedback on their code from the community. Similarly, in contests in which participants cannot work in teams or can only work in much smaller teams individuals do not have as many opportunities to collaborate within their team, thus more collaboration

	% Users Shared Code		Num. Forks	
	Coef.	SE	Coef.	SE
Intercept	0.11	0.06	222.2	163.6
Restricted Teams	0.009	0.018	102.7*	44.5
Prize Amount	0.003	0.004	10.2	12.4
Size Data	0.002	0.002	-11.0	7.0
Contest Length	-0.0009*	0.0003	-1.01	0.86
Num. Participants	-	-	0.11***	0.018
Adj. R^2	0.15		0.78	

Table 3. Linear regression models predicting the effect of contest design effects on the percentage of contest participants who shared code and the degree of collaboration as indicated by the quantity of forked code. The total number of contest participants was included in the second model to control for the popularity of the contest. * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

and sharing may happen outside of teams among the community at large. Together these results suggest that some contest design characteristics that incentivize cooperation and lead to more code sharing and collaboration.

5.3 Research Question 3: Does code sharing affect individual and collective performance in contests?

Sharing code may benefit individuals by providing opportunities for individuals to get feedback, adopt successful approaches, combine approaches and learn from others. We investigated the effect of sharing code on an individual's short-term and long-term performance. We built a mixed-effect negative binomial regression model to test the effect of sharing code on an individual's final points in a contest. We included a random effect for the contest ID and controlled for individuals's performance half way through the contest. The model shows that sharing code is associated with doing better in the contest even after controlling for early performance (Table 4, Model 1).

Similarly we build a negative binomial regression model to test the effect of sharing code on an individual's most recent cumulative points on the site, controlling for their performance in their first contest and the total number of contests that they had entered[7]. Model 2 shows that sharing code in at least one contest is associated with higher long-term performance on the site even after controlling for early performance and overall experience (Table 4). Both of these results suggest that sharing code is associated with better individual short-term performance in a contest and long-term performance over many contests.

There is greater potential for the spread of the best solutions, improvement on the best approaches, and integration of approaches when more individuals share code in a contest. We hypothesized that more code sharing would be associated with better collective performance. We examined collective performance in two ways, the degree to which the best scoring submission improved from the midpoint to the end of a contest and the degree to which the median scores improved from the midpoint to the end of a contest. To avoid potential confounding effects of individuals entering a contest because more code was shared we limited the analysis to individuals and teams who had participated from the beginning of

[7]In order to control for the effect of early performance on later performance we limited our analysis to individuals who had competed in at least two contests

the contest[8]. We found that there was no association between the proportion of users who shared code and the percent improvement in the top score[9] (Table 4, Model 3).

When we examined the effect of more users sharing code on average performance, we found a significant relationship in the opposite direction (Table 4, Model 4). A closer examination of the data showed that this counterintuitive result was due to an unusual contest in which none of the early contestants shared code. In a model without this outlier, there was no longer a relationship between the percentage of users who shared code and improvement in average performance (Coef = -0.53, SE = 1.98, p = 0.79). Thus, the effect of code sharing on average performance is inconclusive for this relatively small data set.

Code sharing does not appear to raise average or top performance in a contest. This result is consistent with our finding that individuals that are doing the best in a contest are less likely to participate in sharing code. In addition, individuals who do share code may not be sharing their best solutions. The very best solutions and approaches may stay hidden and therefore do not have a chance to be improved upon or combined.

5.4 Research Question 4: What code gets modified?

Sharing code can lead to cumulative innovation, in which the community improves, combines, and builds on shared code. Figure 2 shows some examples of code branches that participants have created on Kaggle by forking code. To better understand what code attracts modification and innovation, we applied principles from Podolny and Stuart's [23] study of cumulative innovation using patent data.

First, we predicted that code would be more likely to attract modification if it was written by higher ranked participants and was given a higher community rating because these markers suggest greater legitimacy. Table 5 presents the results of mixed effects logistic regression models using attributes about kernels shared in the first half of the contest to predict whether a kernel is forked in the second half of a contest[10]. We included a random effect for the contest ID. Model 1 shows there is no significant relationship between a kernel author's standing either on the site or in the contest and whether a kernel attracts modification. There is a significant positive relationship between a kernel's community rating and its likelihood of being modified. Our results suggest that unlike Podolny and Stuart's [23] findings, participants are not using author attributes to judge legitimacy. Instead they are using community ratings, presumably because pooled ratings are perceived by the crowd as a better measure of legitimacy and code quality.

Second, we predicted that kernels that were more prototypical would attract modification because they would be more accessible and thus explored first. We computed the average distance between each kernel and the set of other base kernels shared in the earlier contests. Model 1 shows that this distance from the code base is negatively associated with the likelihood that a kernel is modified (Table 5). In other words, kernels that are more unique are less likely to be modified than kernels that are similar to existing code. These findings confirm the second prediction and are in line with findings from Podolny and Stuart [23].

Finally, we predicted that kernels that had only inspired a narrow set of modifications early on would be less likely to attract more modifications later. We measured the average

[8]However, the results do not change if we analyze all individuals and teams who participated in a contest.
[9]The model presented in Table 4 omits one contest in which there was 6×10^{18}% improvement, an extreme outlier, the results are the same when regression is performed on all contests including this outlier.
[10]We consider all forks in a branch family (forks, forks of forks, etc.) to be modifications of a base kernel.

Model 1: Individual Short-Term Performance

Contest Points	Coef.	SE
Intercept	6.62	0.11
Shared Code	0.23***	0.015
Contest Points 1^{st} Half	0.08***	0.001

Model 2: Individual Long-Term Performance

Site Points	Coef.	SE
Intercept	7.61	0.017
Shared Code	0.12***	0.03
Site Points 1^{st} Contest	0.00004***	0.000
Num. Contests	0.01***	0.002

Model 3: Collective Top Performance

% Improvement	Coef.	SE
Intercept	0.29	0.32
% Users Shared Code	-1.66	3.02

Model 4: Collective Median Performance

% Improvement	Coef.	SE
Intercept	0.85	0.26
% Users Shared Code	-6.23*	2.40

Table 4. Models 1 and 2 display the effect of sharing code on individual performance in short and long term respectively, controlling for early performance. Model 1 is a mixed effects negative binomial regression model predicting participants's final contest performance controlling for contest performance in the first half with contest ID as a random effect (39,725 observations and 25 groups). Model 2 is a negative binomial regression model predicting participants's total cumulative points on the site controlling for their performance in their first contest (df = 4866). Model 3 and 4 are linear regression models predicting percentage improvement in maximum and median contest submission scores from the midpoint to the end of a contest respectively based on the proportion of users who shared code (Model 3 $F(1,22) = 0.30$, $p = 0.59$; Model 4 $F(1,23) = 6.71$, $p = 0.02$, Adj. $R^2 = 0.19$). * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

distance between a kernel and its modifications during the first half of the contest[11]. Model 2 shows that that there was a positive association between the average distance within a code branch and the likelihood of the branch attracting a new modification in the second half of the contest. We argue that code that has already produced more distant modifications has more innovation potential because there is more to exploit than code that has at most produced a narrow set of derivative modifications. Our analysis for Model 2 controlled for branch size, and, unsurprisingly, we found that code that had already attracted more modification in the first half of a contest was more likely to attract modification in the second half of a contest.

[11]A second model had to be used for the third prediction because analysis was done on a smaller sample of those kernels that had already inspired forks in the first half of a contest.

	Kernel Forked 2^{nd} Half			
	Model 1		Model 2	
	Coef.	SE	Coef.	SE
Intercept	5.19	1.35	-1.83	0.31
Kernel Author's				
Site Points	-0.000	0.000	-	-
Contest Points 1^{st}	-0.000	0.000	-	-
Community Rating	0.043***	0.008	-	-
Code Base Distance	-6.83***	1.45	-	-
Branch Distance	-	-	1.47***	0.43
Branch Size 1^{st}	-	-	0.39***	0.07

Table 5. Mixed effects logistic regression models testing the effect of kernel characteristics on whether the kernel branch gets expanded in the second half of the contest. Model 1 tests the effect of markers of kernel quality and the average distance between a kernel and the established code base on whether the branch is expanded (1,599 observations, 22 groups). Model 2 tests the effect of the average distance between a kernel and its forks on whether the branch is expanded (271 observations, 24 groups). * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

6 DISCUSSION

Open innovation contests which support community-level collaboration combine two powerful forces for innovation – competition among participants to find the best solution and cumulative innovation through sharing, combining, and integrating solutions. However, they create a tension between sharing and secrecy. The goal of this research was to understand if collaboration could thrive in highly competitive innovation contests. Through four exploratory questions we investigated community-level collaboration on Kaggle in the context of this tension. This study complements and extends three earlier qualitative studies which examined community-level collaboration in open innovation contests [9, 15, 19].

We expected that the competitive nature of innovation contests might inhibit collaboration and that the degree to which collaboration was inhibited might be moderated by the contest's design. We found only 10% of individuals chose to share code during contests on Kaggle and that individuals who were performing the very best in a contest were less likely to share code than similar peers. This is in line with previous research which suggested that some individuals in contests chose to work collaboratively and competitively, while others chose only to work competitively [9, 19]. This study complements prior findings by estimating at what rate individuals choose to collaborate and identifying which type of individuals collaborate. We suspect that there was not more collaboration, and in particular a reduction in relative collaboration from the very top performers in a contest, because individuals wanted to keep code secret to perform better than others in the contests. This finding reflects the general sentiment about sharing code expressed by one user "one of the common critiques of the Scripts is that they allow people to achieve high leaderboard positions with very little effort"[12]. We also examined how contest design affected collaboration and found that contests that restricted team size had more code sharing. We conjecture that when within-team collaboration is not available participants are more likely to be willing to sacrifice secrecy to gain the benefits of collaborating with the community. These results suggest that there is an unrealized potential for more collaboration. Future work, should explore other ways to

[12]https://www.kaggle.com/dvasyukova/scripty-mcscriptface-the-lazy-kaggler

alter contest design to reduce the potentially inhibitive effect of competition and promote community-level collaboration.

Based on prior work on a variety of other online platforms collaboration was expected to be individually and collectively beneficial when it did occur (e.g. [8, 21, 25]). We found that individuals performed better in the short-term and long-term when they collaborated, but that there was no association between collaboration and community level performance. This latter finding is in contrast to [8]. One major difference between contests on Kaggle and [8] which showed that sharing code could increase performance through cumulative innovation is that on Kaggle users decide whether or not to share code where as in the other study all submitted code was automatically shared. We suspect that the lack of more extensive code sharing on Kaggle, particularly by the top performers in the contest, meant that the best code was not shared during contests on Kaggle stifling the potential for cumulative innovation. This finding about collective performance suggests that there is an untapped potential for better collaboration that may necessitate sharing of specific code.

One of the largest potential benefits of sharing code is cumulative innovation, in which individuals share, combine, integrate, and build from each others's work [8]. Gulley and Lakhani [17] find that more collective knowledge is created when individuals write more novel code and recombine borrowed code with novel code. On Kaggle, we observed that shared code was often modified by others. We took research on how patent citation networks grow, an active setting for cumulative innovation, and tested whether there were similar patterns in how code evolved on Kaggle. Code evolved on Kaggle in ways that are consistent with how patent citation networks grow. We found that code that was likely seen as more legitimate, accessible, and exploitable was modified more often. However, promoting modification of code may not be enough to generate better code, new solutions, or substantial improvements over existing solutions. Hill and Monroy-Hernandez [18] found in studying remixing on the Scratch online community that there was a trade-off between the quality and quantity of remixes. The factors that promote more remixes, were the opposite of those that promoted more original remixes. For example, they found that projects that were written by a more prominent author were more likely to be remixed, but that the remixes were less original. The way we observe code evolving on Kaggle may not necessarily be beneficial in terms of producing valuable code that improves the top solutions. Future research is needed to better understand how modifications alter original code, whether modifications improve original code, and whether modifications are instrumental to final solutions submitted to contests.

In this study we found that competitive-collaborative tension was a useful framework for understanding user behavior, the effect of contest design on behavior, and collaborative processes. This framework can enhance existing research on collaboration in open innovation platforms and be used to better design such contests.

7 DESIGN IMPLICATIONS

Organizations hosting contests have a vested interest in getting the best possible solutions to their problems regardless of how those solutions are produced. Prior work on open innovation suggests greater openness and collaboration should lead to the best solutions. However, we argue based on our results that the competitive nature of innovation contests may inhibit this type of open collaboration. Thus, we recommend that platforms take steps to design contests to promote more openness, sharing, and collaboration, particularly from the top performing participants. There are a few approaches platforms could take. They could set up the platform so all code was automatically shared or shared by default (e.g. [8]); they could incentivize collaboration, such as by removing other more private methods of collaboration

like competing as a team; or they could reduce the competitive nature of contests, such as by removing leaderboards that show individuals rankings during the contest. Future work is needed to determine if any of these would be effective without causing other adverse consequences. These same design implications are likely to benefit the platform as well as long as they do not drive away users.

The results of this study suggest that individuals may improve in their short-term and long-term performance when they share code. Yet, individuals on Kaggle often have reservations about sharing code because of how others may use that code ("I can see why people ... would be frustrated about being pushed down in the ratings by the script chasers [users who copy and submit shared code without modification]"[13]). There are two ways platforms could better support individuals. First, they could reinforce intrinsic motivations which are likely to underlie collaboration, such as learning, helping, or belonging to the community. Second, they could provide extrinsic motivations for sharing code, such as implementing an attribution system that recognized performance gains from shared code. Kaggle has some rewards for sharing code (e.g. separate leaderboard, badges) these may not be enough because they are separate from rewards associated with contest outcomes. An attribution system with prizes awarded for final solutions as well as incremental contributions toward the winning solution could improve incentives to share for top contestants and minimize the conflict between collaboration and competition. For example, the use of impact factor metrics, like the h-index, for career advancement in academia encourages cumulative innovation by giving more credit to scholars whose ideas have been incorporated into others' work.

8 LIMITATIONS AND FUTURE WORK

In this study we observed participant behavior on Kaggle. One limitation of an observational study is that we do not have direct insight into participants' mental state, motivations, or perceptions. We proposed several explanations for our observations, such as the very top contestants share less code than would be expected because they do not want to give away their potentially winning solutions. While the observations of behaviors themselves are relevant to the success of collaboration on Kaggle, our explanations of these behaviors may not be correct. This study would be strengthened by future work which uses surveys of participants to match participants' behavior with reasons for sharing or withholding code.

Another limitation of a purely observational study is we cannot determine causality. We found several associations between early behavior and subsequent outcomes, such as early ranking in a contest affects subsequent code sharing and sharing code in one contest affects subsequent performance on the site. By dividing the data into time periods we partially, but not fully, control for the directionality of effects. Future work is needed using more sophisticated quasi-experimental designs and actual experiments to determine directionality and the underlying mechanisms for these effects.

In this study we focused on sharing rather than using shared code. Ultimately, both sharing and using shared code are important in studying cumulative innovation and the impact of shared code on performance, which are both critical areas of future work to assess the value of community-level collaboration. Kaggle did not provide enough information in its database to assess how shared code is used. It does not provide records of who views shared code or records of all submitted code. In exceedingly rare cases individuals submit shared code directly as a submission (and these were recorded), however we suspect these records undercount the actual use of shared code. Future work using interviews and surveys

[13] https://www.kaggle.com/dvasyukova/scripty-mcscriptface-the-lazy-kaggler

is needed to understand the use of shared code and to directly measure how usage impacts performance.

Finally, we examined community-level collaboration on only one site, Kaggle, which hosts a specific type of contest. In many ways contests on Kaggle are similar in design to most other innovation contests–prizes are awarded only to the top few contestants; contestants can share solutions or partial work; and individuals participate for a variety of reasons some extrinsic and some intrinsic. There are two key differences between Kaggle and some other innovation contests that may affect the generalizability of this study's results. First, on Kaggle there is an objective measure of performance that is displayed during contests. This is in contrast to other contests in which performance is more subjective (e.g. design contests) and there is no clear rankings of users or solutions during the contest because judges determine who wins at the end. Second, on Kaggle sharing code is optional; this is very different from contests and platforms that make all submitted code automatically visible to everyone (e.g. [8, 17]). More research is needed to study community-level collaboration on a variety of platforms and types of contests.

9 CONCLUSION

Many innovation contest platforms, like Kaggle, support community-wide collaboration. We found that around 10% of users shared code during contests on Kaggle. Allowing community-level collaboration creates a tension between sharing and secrecy, such that people preferred collaborating within teams rather than at a community level and the very top level contestants collaborated less. As a result, we observed that sharing code helped individuals improve, but did not help improve the winning solutions. Besides providing individual benefits, the potential benefit of community-wide collaboration is cumulative innovation, in which individuals might share, combine, integrate and build on each others's code to provide better winning solutions than individuals or teams could alone. In observing how code evolved we saw similar patterns of collaboration as has been observed in other cumulative innovation settings. Future research should explore the process of collaboration in more detail and test alternative platform designs that might encourage more code sharing particularly of the best solutions from top contestants. Findings from this and future studies should be synthesized into design insights that will help ease the tension between collaboration and competition in online platforms like Kaggle, realizing the full value of open innovation contests.

ACKNOWLEDGMENTS

We thank CSCW editors and reviewers for their remarks which helped improve the paper and Kaggle for providing their data. This work was supported by the National Science Foundation (CISE IIS-1546404).

REFERENCES

[1] Sabrina Adamczyk, Angelika C. Bullinger, and Kathrin M. Moslein. 2012. Innovation Contests: A Review, Classification and Outlook. *Creativity and Innovation Management* 21 (2012), 335–360. DOI: http://dx.doi.org/10.1111/caim.12003

[2] Nikolay Archak. 2010. Money, Glory and Cheap Talk: Analyzing Strategic Behavior of Contestants in Simultaneous Crowdsourcing Contests on TopCoder.com. In *Proceedings of the Conference on World Wide Web (WWW)*. ACM, 21–30. DOI:http://dx.doi.org/10.1145/1772690.1772694

[3] Barry L. Bayus. 2013. Crowdsourcing New Product Ideas over Time: An Analysis of the Dell IdeaStorm Community. *Management Science* 59 (2013), 226–244. DOI:http://dx.doi.org/10.1287/mnsc.1120.1599

[4] Robert M. Bell, Yehuda Koren, and Chris Volinsky. 2010. All Together Now: A Perspective on the Netflix Prize. *CHANCE* 23 (2010), 24–29. DOI:http://dx.doi.org/10.1007/s00144-010-0005-2

[5] James Bennett and Stan Lanning. 2007. The Netflix Prize. In *Proceedings of the Knowledge Discovery and Data Mining (KDD) Cup and Workshop*. ACM, 35–38.

[6] Mark Boons, Daan Stam, and Harry G. Barkema. 2015. Feelings of Pride and Respect as Drivers of Ongoing Member Activity on Crowdsourcing Platforms. *Journal of Management Studies* 52 (2015), 717–741. DOI:http://dx.doi.org/10.1111/joms.12140

[7] Kevin J. Boudreau and Karim R. Lakhani. 2010. The Performance Implications of Institutional "Fit": Field Experimental Evidence on an Innovation Task. *Working Paper* (2010), 1–40.

[8] Kevin J. Boudreau and Karim R. Lakhani. 2015. "Open" Disclosure of Innovations, Incentives and Follow-on Reuse: Theory on Processes of Cumulative Innovation and a Field Experiment in Computational Biology. *Research Policy* 44 (2015), 4–19. DOI:http://dx.doi.org/10.1016/j.respol.2014.08.001

[9] Angelika C. Bullinger, Anne-Katrin Neyer, Matthias Rass, and Kathrin M. Moeslein. 2010. Community-Based Innovation Contests: Where Competition Meets Cooperation. *Creativity and Innovation Management* 19 (2010), 290–303. DOI:http://dx.doi.org/10.1111/j.1467-8691.2010.00565.x

[10] Henry Chesbrough. 2003. *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business School Press, Cambridge, MA.

[11] Edward L. Deci, Richard Koestner, and Richard M. Ryan. 1999. A Meta-Analytic Review of Experiments Examining the Effects of Extrinsic Rewards on Intrinsic Motivation. *Psychological Bulletin* 125 (1999), 627–668. DOI:http://dx.doi.org/10.1037/0033-2909.125.6.627

[12] Indika Dissanayake, Jie Zhang, and Bin Gu. 2015. Task Division for Team Success in Crowdsourcing Contests: Resource Allocation and Alignment Effects. *Journal of Management Information Systems* 32 (2015), 8–39. DOI:http://dx.doi.org/10.1080/07421222.2015.1068604

[13] Samer Faraj, Sirkka L. Jarvenpaa, and Ann Majchrzak. 2011. Knowledge Collaboration in Online Communities. *Organization Science* 22 (2011), 1224–1239. DOI:http://dx.doi.org/10.1287/orsc.1100.0614

[14] Andrea Forte and Cliff Lampe. 2013. Defining, Understanding and Supporting Open Collaboration: Lessons from the Literature. *American Behavioral Scientist* 57 (2013), 535–547. DOI:http://dx.doi.org/10.1177/0002764212469362

[15] Johann Fuller, Katja Hutter, and Rita Faullant. 2011. Why Co-creation Experience Matters? Creative Experience and its Impact on the Quantity and Quality of Creative Contributions. *R&D Management* 41 (2011), 259–273. DOI:http://dx.doi.org/10.1111/j.1467-9310.2011.00640

[16] Connie J. G. Gersick. 1988. Time and Transition in Work Teams: Toward a New Model of Group Development. *Academy of Management Journal* 31 (1988), 9–41. DOI:http://dx.doi.org/10.2307/256496

[17] Ned Gulley and Karim R. Lakhani. 2010. The Determinants of Individual Performance and Collective Value in Private-Collective Software Innovation. *Harvard Business School Working Paper Series* No. 10-065 (2010), 1–40. DOI:http://dx.doi.org/10.2139/ssrn.1550352

[18] Benjamin Mako. Hill and Andres Monroy-Hernandez. 2013. The Remixing Dilemma: The Trade-Off Between Generativity and Originality. *American Behavioral Scientist* 57, 5 (2013), 643–663. DOI:http://dx.doi.org/10.1177/0002764212469359

[19] Katja Hutter, Julia Hautz, Johann Füller, Julia Mueller, and Kurt Matzler. 2011. Communitition: The Tension between Competition and Collaboration in Community-Based Design Contests. *Creativity and Innovation Management* 20 (2011), 3–21. DOI:http://dx.doi.org/10.1111/j.1467-8691.2011.00589.x

[20] Lars B. Jeppesen and Karim R. Lakhani. 2010. Marginality and Problem Solving Effectiveness in Broadcast Search. *Organization Science* 21 (2010), 1016–1033. DOI:http://dx.doi.org/10.1287/orsc.1090.0491

[21] Guo Li, Haiyi Zhu, Tun Lu, Xianghua Ding, and Ning Gu. 2015. Is it Good to be like Wikipedia?: Exploring the Trade-offs of Introducing Collaborative Editing Model to Q&A Sites. In *Proceedings of the Computer Supported Cooperative Work and Social Computing (CSCW)*. ACM, 1080–1091. DOI:http://dx.doi.org/10.1145/2675133.2675155

[22] Henning Piezunka and Linus Dahlander. 2015. Distant Search, Narrow Attention: How Crowding Alters Organizations' Filtering of Suggestions in Crowdsourcing. *Academy of Management Journal* 58 (2015), 856–880. DOI:http://dx.doi.org/10.5465/amj.2012.0458

[23] Joel M. Podolny and Toby E. Stuart. 1995. A Role-Based Ecology of Technological Change. *The American Journal of Sociology* 100 (1995), 1224–1260. DOI:http://dx.doi.org/10.1086/230637

[24] Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. 2011. An Assessment of Intrinsic and Extrinsic Motivation on Task Performance in Crowdsourcing Markets.

In *Proceedings of the International Conference on Web and Social Media (ICWSM)*. AAAI, 17–21. DOI:http://dx.doi.org/10.13140/RG.2.2.19170.94401

[25] Yla R. Tausczik, Aniket Kittur, and Robert E. Kraut. 2014. Collaborative Problem Solving: A Study of MathOverflow. In *Proceedings of the Computer Supported Cooperative Work and Social Computing (CSCW)*. ACM, 355–367. DOI:http://dx.doi.org/10.1145/2531602.2531690

[26] Theresa Velden. 2013. Explaining Field Differences in Openness and Sharing in Scientific Communities. In *Proceedings of the Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*. ACM, 445–457. DOI:http://dx.doi.org/10.1145/2441776.2441827

[27] Eric von Hippel and Georg von Krogh. 2003. Open Source Software and the "Private-Collection" Innovation Model: Issues for Organization Science. *Organization Science* 14 (2003), 209–223. DOI: http://dx.doi.org/10.1287/orsc.14.2.209.14992

[28] Yang Yang, Pei-yu Chen, and Paul Pavlou. 2009. Open Innovation: Strategic Design of Online Contests. In *Proceedings of the International Conference on Information Systems (ICIS)*. AIS, 1–42. DOI: http://dx.doi.org/10.1145/2901739.2901772

[29] Lixiu Yu, Paul André, Aniket Kittur, and Robert Kraut. 2014. A Comparison of Social, Learning, and Financial Strategies on Crowd Engagement and Output Quality. In *Proceedings of the Computer Supported Cooperative Work and Social Computing (CSCW)*. ACM, 967–978. DOI:http://dx.doi.org/10.1145/2531602.2531729

[30] Lixiu Yu and Jeffrey V. Nickerson. 2011. Cooks or Cobblers? Crowd Creativity through Combination. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, 1393–1402. DOI:http://dx.doi.org/10.1145/1978942.1979147

[31] Alexey Zagalsky, Carlos Gomez Teshima, Daniel M. German, Margaret-Anne Storey, and Germán Poo-Caamaño. 2016. How the R Community Creates and Curates Knowledge: A Comparative Study of Stack Overflow and Mailing Lists. In *International Conference on Mining Software Repositories (MSR)*. ACM, 441–451. DOI:http://dx.doi.org/10.1145/2901739.2901772

[32] Haichao Zheng, Dahui Li, and Wenhua Hou. 2011. Task Design, Motivation, and Participation in Crowdsourcing Contests. *International Journal of Electronic Commerce* 15 (2011), 57–88. DOI: http://dx.doi.org/10.2753/JEC1086-4415150402

[33] Wenjun Zhou, Wangcheng Yan, and Xi Zhang. 2017. Collaboration for Success in Crowdsourced Innovation Projects: Knowledge Creation, Team Diversity, and Tacit Coordination. In *Proceedings of the Conference Hawaii International Conference on System Sciences (HICSS)*. IEEE, 381–390.

Received June 2017; Revised August 2017; Accepted November 2017