

How Interest-Driven Content Creation Shapes Opportunities for Informal Learning in Scratch: A Case Study on Novices' Use of Data Structures

Ruijia Cheng
University of Washington
Seattle, Washington, USA
rcheng6@uw.edu

Sayamindu Dasgupta*
University of North Carolina at
Chapel Hill
Chapel Hill, North Carolina, USA
sayamindu@unc.edu

Benjamin Mako Hill
University of Washington
Seattle, Washington, USA
makohill@uw.edu

ABSTRACT

Through a mixed-method analysis of data from Scratch, we examine how novices learn to program with simple data structures by using community-produced learning resources. First, we present a qualitative study that describes how community-produced learning resources create archetypes that shape exploration and may disadvantage some with less common interests. In a second quantitative study, we find broad support for this dynamic in several hypothesis tests. Our findings identify a social feedback loop that we argue could limit sources of inspiration, pose barriers to broadening participation, and confine learners' understanding of general concepts. We conclude by suggesting several approaches that may mitigate these dynamics.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; **Empirical studies in collaborative and social computing**; **Collaborative and social computing theory, concepts and paradigms**; *Computer supported cooperative work*; • **Social and professional topics** → **Computing education**.

KEYWORDS

online communities, informal learning, learning to code, connected learning, computational participation, computers and children, creativity support tools, social computing and social navigation, forums, Scratch

ACM Reference Format:

Ruijia Cheng, Sayamindu Dasgupta, and Benjamin Mako Hill. 2022. How Interest-Driven Content Creation Shapes Opportunities for Informal Learning in Scratch: A Case Study on Novices' Use of Data Structures. In *CHI Conference on Human Factors in Computing Systems (CHI '22)*, April 29-May 5, 2022, New Orleans, LA, USA. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3491102.3502124>

*Dasgupta was at the University of North Carolina at Chapel Hill when this work was submitted for review. Since then, he has moved to the University of Washington.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-9157-3/22/04...\$15.00

<https://doi.org/10.1145/3491102.3502124>

1 INTRODUCTION

Scholars increasingly look to interest-driven online communities as promising environments for supporting learning [7, 38, 42]. These communities rely on user engagement in content creation to curate community-produced learning resources where users engage in sharing artifacts that they create and online discussion with other users. Although such communities exist in a range of domains like creative writing [9], graphical design [56], and more, many of the largest and most studied have focused on supporting young people in computational learning (i.e., learning about computational concepts, often through learning to program a computer). Widely studied examples include the Scratch community [59] and MOOSE Crossing [62]. These interest-driven computational learning communities are built around the idea of “computational participation” [45], which encourages learners to develop programming skills through creating and sharing projects and interacting with other learners [6, 7, 44].

Despite this promise, it remains unclear how the creation and usage of community-generated learning resources support the learning of computational concepts. Previous studies of the Scratch online community show that users who remix, or build projects from code shared by other users, achieve better learning outcomes [18]. However, most users do not demonstrate much innovation in remixed projects, making people question whether they actually develop transformative abilities [33]. In fact, most Scratch users only display a limited range of programming skills [57, 79]. For example, only around 15% users have ever used data structures—an important computational concept—in Scratch projects [18]. While online discussion provides opportunities for learners to mentor each other and collaboratively debug programs [24, 69], it can also end up as superficial socialization in ways that can even act as a barrier to the exchanges of ideas, feedback, and resources [69]. These mixed signals suggest the need of a better understanding about the mechanism of interest-driven content creation in computational learning communities. How does computational learning happen through interest-driven content creation? What is the role of community in the process? How do community-produced learning resources support learners' diverse interests?

To explore these questions, we present two studies about the Scratch community that describe the opportunities and challenges that interest-driven content creation and related community activities introduce to computational learning. In Study 1, where we present a grounded theory analysis of 400 discussion threads in the Scratch forums about how learners develop an understanding

of data structures—variables and lists. Based on this analysis, we hypothesize a social feedback loop where engagement in content sharing and Q&A naturally raises the visibility of some particular ways of using variables and lists. Through their increased visibility, these examples become archetypes that can limit the breadth of future projects in the community. In Study 2, we conduct a quantitative analysis of the code corpus of more than 200,000 Scratch projects to test our hypothesis and find statistical support for the social process theorized in our first study. We conclude with several implications for design and content curation that we believe could improve support for diverse interests in Scratch and similar interest-driven learning communities.

This work makes several contributions to the HCI and social computing literature on computational learning. First, we make an empirical contribution by presenting detailed qualitative and quantitative evidence about how novices learn to use data structures in the Scratch online community. Second, we offer a theoretical contribution in the form of a framework describing a dynamic process where interest-driven content creation can both assist learning about particular topics while posing important limits on the ways that those topics are engaged with. Finally, we make a contribution to the literature on the design of informal learning systems by speculating about how online interest-driven communities can be designed to mitigate the negative repercussions of the dynamic we describe.

2 BACKGROUND

Online communities are increasingly common settings for participatory, interest-driven, and community-supported learning [7, 30, 41, 42]. A broad range of theoretical frameworks have been used to design and analyze these communities—many building on foundational theories on the social origins of learning by Vygotsky [75] and Lave and Wenger [52]. Another key theoretical framing is Papert [60]’s view of learning as the construction of knowledge that “happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity” (p. 1) and which emphasizes the importance of interest-driven exploration and “personally powerful ideas” in promoting learning [61]. Recent scholarship on connected learning [39, 40] have also endorsed the role of shared interest and participatory culture in building learning communities.

Learning experiences in interest-driven online communities happen through two primary pathways: through sharing creative artifacts like fan fiction [9], design mock-ups [15], interactive computer programs [17]; and through social interactions around these artifacts like commenting, remixing, and critiquing. To promote the first pathway, many communities are structured so that the products of learning are made visible as public artifacts that can be used by others as illustrative examples and for inspiration [18, 29, 56] as well as scaffolds for replication, practice, and innovation [18, 71]. To promote the second pathway, communities feature direct user-to-user support such as comments [9], forum posts [48], and Q&A discussions [72] that can help members gain an understanding of specific topics or techniques [69]. The two pathways are deeply interwoven. By making artifacts publicly visible, creators are able to receive constructive feedback that can support learning [29, 80].

This often includes input from experts and professionals that is otherwise unavailable [36, 49] as well as social recognition and support [9]. Additionally, learners in online communities often center their social interaction around discussions of public artifacts and as social interactions support the further production of artifacts [46], collaborative problem-solving [53, 69, 72] and community-wise knowledge advancement [31].

In recent years, creative programming communities have emerged as a prominent example of online interest-driven learning communities. In what Kafai [44] has described as “the social turn in K–12 computing” (p. 27) and “computational participation,” scholars have turned to interest-driven and socially supported contexts to promote learning about computing where learners create programs to be shared with peers. Through the work of efforts inspired by this approach, millions of young people have engaged in programming in online communities. Some notable examples of these communities include programmable multiplayer game environments [7], platforms for interactive media creation [59, 63, 77], and amateur technical support groups [25].

Despite the promise of these communities, it remains unclear how to best promote computational participation. Recent studies of online programming communities have shown unequal outcomes in terms of both participation and learning based on gender and race [23, 29, 65] and that important debugging or collaborative sense-making activities are not always helped by socialization [69]. Furthermore, while online communities allow users to gain inspiration from examples posted by others, studies on remixing activities indicate it can negatively impact originality [33]. These examples indicate a lack of a general understanding of the dynamics around learning in online informal contexts. In interest-driven communities of all types, learning pathways can be blocked by the difficulty of ensuring high quality content [1, 36, 47, 56, 78] and the challenge of engaging diverse groups of users [8, 14]. This mismatch between the theoretical promise of online interest-driven communities and what is seen in practice indicates a lack of understanding of why user engagement supports or fails to support learning and how we can best design to facilitate positive learning outcomes.

Because computational participation involves learning many different concepts, we focus on learning experiences over one specific computational concept that has been the subject of substantial academic work: the simplest data structures comprising scalar variables and lists. We consider data structures specifically because Brennan and Resnick [5] identify the ability to understand how to store, retrieve, and update data as one of seven major practices that contribute to computational thinking [3, 21, 76]. Despite their importance, research has shown that data is the least commonly engaged computational thinking concept in Scratch [18]. Previous work by Dasgupta et al. [18] estimates that less than 15% of Scratch users will ever make projects using data structures. When used, it is often engaged with in superficial ways [4].

Why are data structures hard to learn? An explanation stems from the fact that learning about a computational concept involves learning its structural and functional uses [22]. The *structural uses* of variables and lists (i.e., how to integrate them in a program) are straightforward. For example in Scratch, there are only two methods (`get()` and `set()`) that represent the structural usage of scalar variables. However, the *functional uses* of variables and

lists (i.e., what meaningful outcome that they can help create) are both broad and invisible. For example, while variables can be used for storing user input, keeping track of internal program state, counting in a loop, and so on, these functions may not be immediately obvious to novices. Previous work has argued that learners require “specific tutoring” [22, p. 207] and has demonstrated that scaffolds are required to cope with misconceptions about how variables can be used to achieve concrete functionality [32]. It is unclear where interest-driven communities like Scratch are effective in providing proper tutoring needed for learning the functional uses of data structures. Therefore, we ask the following research question: *When does interest-driven content creation most effectively support learning of functional uses of variables and lists? When does it fall short?* We seek to answer these questions through two closely linked empirical studies that unpack practices, challenges, and opportunities for learning about variables and lists in the Scratch online community.

3 EMPIRICAL SETTING

Scratch is a visual, block-based programming language designed for children [63]. Scratch programming primitives are represented by visual blocks that control the behavior of on-screen graphical objects called sprites. Scratch programs (commonly called projects) are constructed by dragging and dropping blocks together. As a programming language, Scratch supports basic data structures in the form of scalar variables and vector lists (Figure 1a).

Primitives to operate on variables and lists fall under the category of “data blocks” in Scratch and their design are described in detail by Maloney et al. [54]. When creating a variable or list, Scratch users assign a name to the variable or list through a free-form text entry field that is invoked through a “make a variable” or a “make a list” button in the Scratch user interface. We refer to these user-defined names as “variable name” or “list name” in this paper. When referring to the variable or list within the Scratch program, users select the appropriate name from a dropdown list embedded in the block. Variables in Scratch have two forms which share a grammar: (1) conventional variables and lists which are local to each instance of a running project; and (2) cloud variables which are persistent across multiple executions of a project and shared across users [16]. Cloud variables are only accessible to established members in the Scratch online community as established by an undocumented set of criteria [20].

Scratch is situated within an online community where anyone can sign up and share their projects, comment on, “like,” and bookmark others’ works, and socialize in forums [59].¹ As of 2021, the Scratch online community has over 65 million registered users, and over 68 million shared projects that span a diverse range of genres and themes. The large majority of Scratch users are between 8–16 years old and the median age for new contributors is around 12.² Although our data might include adults, we follow other scholarly accounts and refer to Scratch users as “kids” [e.g., 38]. We draw data from both the Scratch community itself and its discussion forums, shown in Figure 1b. These forums comprise a number of topical forums organized into categories such as “Making Scratch Projects”

and subcategories such as “Help With Scripts” or “Project Ideas” [68].³

4 RESEARCH ETHICS

This research relies on two sources of data and included no intervention or interaction with subjects. Although the data in our qualitative analysis (§5) are public posts in the Scratch forums, we sampled from these posts using keyword searches of a copy of the Scratch Forums database in a way that the public could not do easily. We have obscured users’ identities by replacing usernames with alphanumeric identities and by following advice from Markham [55] to reword quotes to make it more difficult to identify the Scratch users we quote using simple web searches. Our quantitative analysis of the Scratch code corpus in §6 relied on data that the Scratch team has published as part of the *Scratch Research Dataset* [34]. This work was reviewed and overseen by the IRB at MIT as part of a broader protocol covering observational studies of Scratch. Our institutional IRBs delegated oversight of the work on this project to the MIT IRB.

5 STUDY 1: THEORY DEVELOPMENT

Because we could not identify existing theories with clear prediction on when Scratch would best support learning about variables and lists, we began with an open-ended interpretive analysis that sought to build a theory about kids’ practices, learning, and engagement in discussions about simple data structures.

5.1 Methods

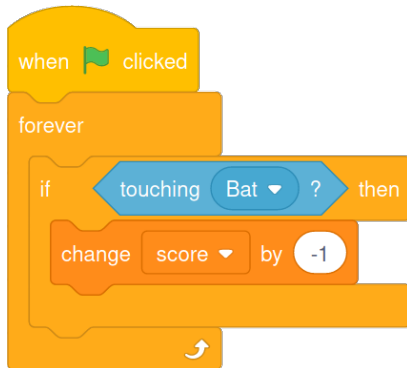
To build a dataset for our qualitative analysis, we generated a sample of 400 discussion threads about variables and lists from the Scratch discussion forums. Because we were interested in how kids learn to make projects with variables and lists by engaging in learning resources curated in discussions, we limited our sample to two subforums that emphasized question asking: *Help With Scripts* and *Questions about Scratch*. We chose to study forum threads instead of the more widely used project comments because previous work suggested that only a small number of comments were related to problem solving [69]. The dataset that we used for sampling contains discussions that took place between October 11, 2012 and April 5, 2017.

To acquaint ourselves with our setting, we spent several weeks browsing the forums. As part of this process, we found that the many posters of threads about data structure did not use the specific terminology of “data block,” “variable,” or “list” in their posts. Therefore, to include a broad range of conversations about data in the Scratch forums, we followed advice from Trost [74] to build a “statistically non-representative stratified sample” that would ensure that a range of different ways of talking about data were reflected in our data, but without concern for each type’s prevalence. To do so, we sampled threads in three stages using different keywords. First, we randomly sampled 100 threads from titles containing the keywords “variable,” “list” or “data.” Second, we sampled 10 random threads each from the 11 most common variable or list names—as identified in an analysis of the Scratch code corpus by Dasgupta [17]—for 110 threads total. These terms included “high

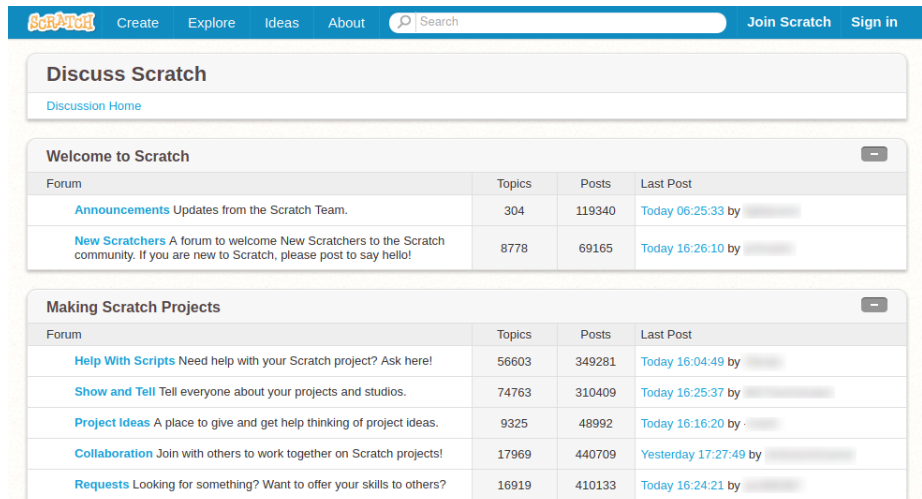
¹<https://scratch.mit.edu/>

²All statistics about Scratch community activity and users are taken from the public information on: <https://scratch.mit.edu/statistics/>

³<https://scratch.mit.edu/discuss/>



(a) Sample Scratch code showing a variable in the form of a data block called “score” being decremented on collision with a bat sprite



(b) Index page of the Scratch forums

Figure 1: The Scratch programming language and online community

scores,” “lives,” “inventory,” “leaderboard,” “speed,” “timer,” “counter,” “words,” “points,” “velocity,” and “answers.” Third, to further increase diversity in our dataset, we randomly sampled two threads with each of the other top 100 variable and list names. This step resulted in an additional 200 threads. We only included threads with more than a single post because we wanted to ensure that the thread contained at least some form of a discussion. All together, these sampling steps resulted in 410 threads. Because 10 threads were included in more than one of our samples, we ended up with a total of 400 threads that contained 2,790 posts and 963,593 words of content—equivalent to 547 pages of single-spaced text.

We analyzed these data using Charmaz’s [13] approach to constructing grounded theory. In the open coding phase, the first author led the analysis and started by annotating the threads with both line-by-line and incident-by-incident codes. In the process, the first author regularly shared coded data with the rest of the team and iteratively update the codes based on team discussion. For example, in vivo codes were generated about what kids were creating using data structures: “making a score counter,” “making a leaderboard,” and so on. Following Charmaz, a small number of “sensitizing codes” drawn from existing theories were also used in the open coding phase, such as “engaging in collaborative debugging,” [69] “helping through remixing,” [18] and so on. In the axial coding phase, the first author led the process of code development by grouping initial codes into themes and meta-themes. For example, the initial codes about what kids were creating using data structures were grouped into an axial code called “functional use of data,” with sub-level axial codes on variables and lists. This process involved repeated team discussions on code development, several rounds of iteration on code schemes, and re-coding data. Lastly, we composed memos to describe connections among the axial codes. Our findings reflect the content of the memos written about the major themes that emerged at the end of our analysis.

5.2 Findings

Our analysis resulted in three major themes. First, we discovered evidence that learners, driven by an interest in making specific popular game elements, tend to adopt a narrow set of functional uses of variables and lists. Second—and as a result of the first finding—we found that user-generated learning resources about variables and lists are framed around those specific examples of functional uses. Finally, we identified that those specific examples become archetypes that restrict the breadth of future functional uses in the community. Together, these findings describe a grounded theory about how interest-driven content creation can limit learning opportunities.

5.2.1 Learners create projects with functional uses of variables and lists specific to their interests in game-making. We found that Scratch users were often introduced to variables and lists when they engaged in discussions about specific functional elements of their projects. Because kids in our sample usually had specific goals for their projects in mind, but little knowledge about how to realize those goals in code, they would describe the particular thing they wanted to do when seeking help. The concepts of variables or lists would typically be brought up by someone else in response. Although a quarter of our sample were not selected on variable names and half of our sample included threads based on 100 different variable names, game-making was an almost ubiquitous topic of discussion. Over and over, we observed kids phrasing their questions in terms of game-related goals in discussions in which variables and lists were eventually brought up. In the following sections, we discuss the canonical game-related use cases for variables, lists, and cloud variables in turn.

Variables: Two examples of common game elements that kids like to make in Scratch are *score counters* and *animations*. Score counters are a game element that keeps track of a player’s score or remaining lives. Although this game element is broadly familiar to Scratch users, a user seeking to implement a score counter for the

first time may not know about variables. Indeed, it is not always even obvious to users who know about the existence of data blocks in Scratch that a variable is an appropriate way to keep track of a changing quantity. Furthermore, it can be challenging for kids to implement a counter and integrate it into their program. For example, K1 asked: “I would like to know how to add lives in my game. I want it so that whenever the main character touches a ghost, it would lose one life.” In interactions like these, other users would introduce variables and their functional use as a score counter. In this case, a reply from K2 suggested K1 “create a variable with the name lives” and use it to control the visual elements that represent the character’s lives. Although carefully scoped to the specific problem faced by K1, the reply highlighted the role of variables in tracking changes.

Another common pathway to learning about variables involved animated objects in games. Animation frequently relies on variables because it involves changing the speed or size of objects when triggered by conditions. For instance, K3 asked:

I’m making a pong game where I want to add a control to tell the ball to go faster. Is there a button for this? If not, how can I make this work?

K3 arrived to the Scratch forums knowing that they wanted to vary the speed of a pong ball. For them, the challenge was the specific case of making a ball move at a range of different speeds. Responding to their question, K4 pointed to variables:

Somehow, you must tell the ball to move. Make sure you use a variable. Like use ‘move [speed] steps’ rather than ‘move 5 steps’. Then you just need to set your variable to the speed you like.

This response suggested that the K3 use variables and told them how. In these examples and many others, we saw that variables—both the concept and the term—almost never appeared in learners’ initial questions. Instead, many of the Scratch forum’s learning resources about variables existed in answers to questions about score counters, animation, and other game elements.

Lists: We found a similar pattern for the list data structure. Frequently, kids were introduced to list data structures when trying to figure out how to make inventories—a game element with which players can store and retrieve items. For example, K5 asked: “Does anyone have an idea how to make a good inventory for a game?” K6 answered: “You can use the list block to store your items in the inventory.” In another example, K7 said, “Allllllright so I’m making an inventory for game (who wouldn’t want one?) so I don’t know how to make one. Can anybody help?” These kids were all pointed to lists. Discussions like these helped kids who were struggling to build inventory features connect the abstract concept of a list with its functional use of storing multiple game items and played out repeatedly in the forums.

As with variables, kids imagined how an inventory would be used in the context of their games. For example, they described backpacks of weapons or a pool of correct answers in a quiz game. These kids also imagined rules describing how a player should manipulate items in the inventory. For example, K8 wanted to make a weapon inventory to hold “basic armour” and hoped to “make the player lose less blood when he/she has those items.” After sharing these ideas, they received suggestions to put a list data

structure within an “if else” statement. Cases like this suggest that building inventories allowed kids to learn not only about how to populate and read from lists, but also about basic list operations like deleting and appending items and about conditions and loops. In some cases, more advanced learners would describe methods for accomplishing complicated tasks imagined by novices:

K9: I am making a game where you can buy food and eat it. I want it so you can delete a certain food item from a list... so when you click, the sprite called ‘Strawberry Popsicle’ disappears, but also the name ‘Strawberry Popsicle’ disappears from the list too.

K10: You need to search in the list and find the item that you want to delete. You can just look at each item in the list and compare it to the one you are looking for. Then you stop when you find it or get to the end of the list. This is called a sequential search. [Example code to solve the problem]

This thread shows how relatively sophisticated algorithms were explained in terms of very specific use cases, often with example code. By exchanging ideas about inventories in games, kids introduced each other to lists, their function, and the way they could be used.

Cloud Variables: A final example extends this pattern to *cloud variables* (see §3). Cloud variables’ ability to store data in ways that are persistent and shared were essential for users building “leaderboards” or “high score” systems that could record, rank, and display scores from multiple players—e.g., “a leaderboard in which the highest scores of every player of the game could be saved” (K11). Discussions about leaderboards often involve pointing out the existence of cloud variables, the differences between local and cloud data, and ideas about how to write code to use both. These conversations often segued into advanced programming topics like the encoding and decoding of strings. As with variables and lists, these conversations typically remained focused on the specific use case of leaderboards.

In all three cases, specific use cases became linked to specific data structures—variables with counters and animations, lists with inventories, and cloud data with leaderboards. Because questions tended to focus on these types of elements, discussions about solutions did as well. Through this process of user-to-user support, kids learned how to apply data structures in an informal and unstructured manner. As we describe in the next section, both these conversations and the games that Scratch users created acted as learning resources about variables and lists that were subsequently used by other learners in the Scratch community.

5.2.2 User-generated learning resources about variables and lists are framed around specific examples of functional uses. One feature of informal online learning environments is that conversations and solutions act as learning resources for subsequent participants facing similar challenges. In ways that are visible in our examples in §5.2.1, both the questions posed and the answers provided in our sample tended to focus on specific game-related functional uses. As a result, learning resources about how to use variables and lists tended to be framed in terms of specific game elements and rarely engaged with the more general concepts about data structures. For

example, the following threads show a discussion on how to change the speed of a ball using variables:

K13: Can someone help me figure out how to change the speed of the ball when it hits the paddle a certain number of times?

K14: You can make a variable called HITS, set it to 0, change it by 1 every time you hit the paddle. When it gets to the number you like, change costume and set counter to 0.

In this exchange, K14's answer details exactly what K13 should do to solve their problem—specified down to the names of variables. While this answer likely solved K13's problem, it is anchored on a very specific functional use of variables and did not explain, or suggest that there existed, other potential functional uses. Discussions like this produce learning resources that are extremely specific to the question askers' use case.

When helping others use variables, kids would often refer to popular or example projects that contained working code. For instance, K15 asked “how to make a counter for scores using a sprite making clones of itself” and was directed by others to an established code chunk “changeScore method” in an existing project. In some cases, kids with more advanced knowledge would post snippets of working code:

K16: So I have a chat game, when “hello” is clicked, the robot would say “hello”. I wonder how to make the robot say like a option of things such as “hey” or “yo” instead of “hello” all the time?

K17: Put all the hello, hey words in a list then use this code: say (item (random v) of [list v])

Although these are wonderful examples of kids mentoring each other, the solutions often offered by kids in our data were so specific—and almost always related to game elements—that they would be unlikely to help a novice learner build a conceptual understanding of data structures. It is not hard to imagine that, if a kid with a specific problem solvable with variables were to browse the discussion threads we analyzed, they might not be able to understand how variables could solve their problem unless they were making a game that was similar to one made by a person who had posted a question.

Furthermore, kids offered code might be able to use it without understanding it [67]. We saw many examples of kids requesting working solutions and many others who seemed happy receiving code that could be copy-and-pasted into their programs. For instance, K18 requested help in the form of an insertable code block: “Does anyone know how to make a smooth jump script? If you can make it into a custom block that would be great.” In another example, K19 described the specific effect they wanted and expressed hope that someone could write the code for them:

I want to have a list that has these items: ham, cheese, egg, butter... I need it to find egg and read out it's number in the list. Is there a working script for this?

As requested, K19's post was followed by a code snippet with variables named as K19 imagined them.

These examples are part of a broader pattern. To support kids like K18 and K19, the Scratch community creates solutions that are

directly applicable to particular use cases. While these responses help kids overcome their problems quickly, directly workable solutions mean that kids might not see the broader picture of how variables and lists can be used. We explain in the next section how, because learning resources in communities like Scratch consist of questions and solutions accumulated over time, this high degree of specificity in knowledge resources can result in difficult learning experiences for some.

5.2.3 Specific examples become archetypes and limit the breadth of functional uses in the community. Because learning resources were framed around specific examples of functional uses, many of the Scratch users in our forum dataset appeared to have a limited understanding of what variables and lists could do. This restricted understanding meant that even users without an expressed interest in making games, or particular elements in games, were presented with resources based on them. For example, a user expressing curiosity about data blocks in very open-ended terms received an answer that based on score counters:

K20: I think data blocks can be useful in my projects, but I don't know how to use them.

K21: You are saying variables and lists? For variables, they are just ways to name and store things. So if you make a game and want to keep the score then you'd create a variable called score.

Although K20 did not express any interest in games, K21's response was focused on them.

This reliance on canonical use cases was particularly obvious in discussions about more advanced cloud variables. For example, in the beginning of the following thread, K22 stated that they did not have knowledge about cloud variables. Despite lack of knowledge on the concept, K22 directly pointed to a canonical use case of it that they had heard of, that is, multiplayer games with leaderboards and high score lists. Immediately after this post, two other kids (K23 and K24) started a discussion thread about the particular use case:

K22: I have no idea what cloud variable is but I heard you could make multiplayer games with it.

K23: They are shared by 2+ instances of your game. If you make a very simple game in which you add 1 to a cloud variable when a sprite is clicked. Save your game. Then open the game in two new windows in your browser.

K24: They are usually for High Scores and Multiplayer Games.

Although K22 signalled that they were open to exploration, the answers they received from K23 and K24 indicated the most canonical goals and interests around cloud variables.

We found that kids with interests that deviated from canonical use cases had difficulties finding learning resources that fit their interests. For instance, when K25 posed a very general question about “how to use cloud variables to save data from users” in a more general thread about cloud data, other kids tried to help by posting examples of a high score system that involves cloud variables. K25 expressed confusion because the solution for a high score system did not fit their own goals and said, “but my game isn't one of those

scoring games. I want to make a storyline game.” K25’s comment revealed what much of the Scratch forums community takes for granted. K25 ended up not receiving help in the thread. Over time, their post was lost and ignored in a stream of more typical messages about leaderboards.

In summary, we found that certain functional uses became the Scratch forum’s go-to examples for explaining variables and lists. Because learning resources were built cumulatively, it is not hard to imagine that more projects with these functional uses of variables and lists would be created over time. These new projects, in turn, became new learning resources as well. Learners who wanted to explore different use cases had fewer relevant resources to guide them.

5.3 Synthesis: A theory of social feedback loops in interest-driven online learning communities

Our findings echo Papert’s [61] emphasis on “personally powerful ideas” and Ito et al.’s [39] description of interest-driven, community-based learning. We found that kids in Scratch are motivated to use variables and lists to explore their passions and that they leverage content shared by others in the community to do so. Because learners are working with specific goals in mind, they run into the need for variables and lists while trying to implement specific elements. They are tutored by peer-produced learning resources framed in terms of those specific functionalities. In this sense, our finding contributes to both the literature of computational participation and computing education by describing that interest-driven content creation can be a potential pathway to support learning of functional uses of complex computing concepts.

At the same time, we also discovered a unintentional side effect of this type of learning. We identified that because community-generated learning resources in the form of Q&A, tutorials, and project examples tend to be directed toward specific functional uses, those that represent common interests can become archetypes in ways that leave less room for unconventional interests. In some cases, it can also lead to a shallow understanding of underlying concepts [67]. In our sample, the almost exclusive focus on certain game elements raises concerns about whether learners who are not interested in making these elements will be well served by community-generated resources. Furthermore, we observed that kids who were open to exploring different functional uses are pointed to the archetypal use cases. Overtime, this practice can make the most common use cases even more archetypal.

Inspired by the existing body of literature on network-based and social processes that lead to increased concentration of resources over time, such as the Pareto principle [43], preferential attachment [2], and the Matthew effect [58], we theorize that this process plays out as a *social feedback loop*. Our social feedback loop theory is situated in the specific context of interest-driven computational learning in Scratch and can be summarized as follows: *interest-driven content creation can result in certain types of creation becoming archetypes that make community-generated learning resources more homogeneous and support an increasingly limited set of learner interests over time.*

This theory is visualized in Figure 2. Stage 1 suggests that some types of creation (Use Case A) will be more popular than others and that more users will tend to create artifacts with Use Case A than other use cases. This may be due to initial user base homogeneity, targeted recruiting in the beginning of the community, examples used in documentation, and so on. The arrow pointing from Stage 1 to Stage 2 captures the process of learners running into problems and seeking community support. We argue that users will tend to ask questions framed specifically around Use Case A and draw inspiration from others’ artifacts with Use Case A. Stage 2 shows the results of this process. As learners successfully receive support, they produce new artifacts with Use Case A that can serve as learning resources for others. The arrow on the top of Figure 2 pointing from Stage 2 back to Stage 1 shows how subsequent learners draw inspiration and support from these accumulated learning resources and, as a result, become even more likely to create artifacts with Use Case A in the future. The box on the bottom of Figure 2 captures the outcome of the feedback loop based on our findings from §5.2.3. As Use Case A becomes an archetype, the community’s collective learning resources are increasingly focused on Use Case A as well. Learners like K25 in §5.2.3 who have different interests receive less support. Over time, innovative use cases will become less prevalent.

6 STUDY 2: THEORY TESTING

Study 1’s ultimate findings are a set of untested propositions. In a quantitative follow-up study, we conduct tests of three hypotheses that we derived from the theoretical model presented in §5.3 to begin the process of validating the theory. Our first two hypotheses (marked as “H1” and “H2” on the bottom of Figure 2) focus on the outcome of the feedback loop, that is, what we will observe if we assume the social feedback loop is occurring. In both cases, the hypotheses are that such a feedback loop will make use cases increasingly homogeneous over time. First, we hypothesize that certain genres of projects involving simple data structures will become more popular over time relative to other genres. Based on our findings in Study 1, we hypothesize that **(H1)** *over time, more projects involving variables and lists will be games.*

Second, we hypothesize that popular functional uses of variables and lists will be even more common relative to others over time. In Scratch, users are able to enter a free form string as the name of their variables and lists. Based on our observations in Study 1, users tend to name variables and lists as the specific things they are trying to make. Therefore, we treated the names that users assign to the variables and lists as a proxy to the functional uses. We thus hypothesize that **(H2)** *the names that users give to variables and lists will become more concentrated over time.*

While these hypotheses reflect what we would expect to see in aggregate if the hypothesized social feedback loop were occurring, our third hypothesis (marked as “H3” on Figure 2) attempts to capture part of the theorized mechanism, that is, users who get exposed to archetype use cases will create similar artifacts. Therefore, we hypothesize that **(H3)** *users who have been exposed to projects involving popular variable and list names will be more likely to use those names in their own projects compared to users who have never been exposed to such projects.*

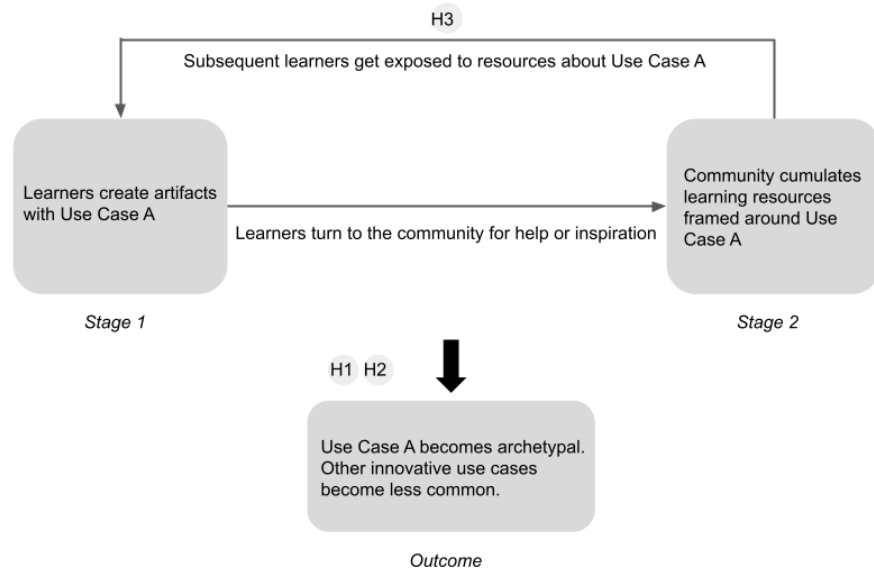


Figure 2: Hypothesized social feedback loop in interest-driven online learning communities

6.1 Data

To conduct our quantitative analyses, we used the *projects*, *project_strings*, *project_text* tables from the publicly available Scratch Research Dataset [34]. For testing H3, we utilized one non-public column that records which users had downloaded others’ projects. We restrict our analysis to projects created between September 2, 2008 and April 10, 2012 because affordances around data blocks were consistent during this period.⁴ The period is earlier than the time window used in Study 1 based on differences in the datasets we had access to. For analytical simplicity, we decided to only include projects with variables and lists written in English. Finally, we restricted our analysis to *de novo* (i.e., non-remix) projects. This resulted in 241,634 projects that contained one or more variables authored by 75,911 Scratch users, and 26,440 projects that contained one or more lists authored by 12,597 users. We created both project-level and user-level datasets with a range of metadata available in the Scratch Research Dataset [34]. In the spirit of open science, we have placed our full source code for dataset creation and analysis into a public archival repository.⁵

6.2 Analysis and Measures

To test H1, we used our project-level dataset to assess whether there is an increase over time in the proportion of games with at least one variable/list. To ensure that our assumption of games being a predominant genre of project was correct, we randomly subsampled 100 projects with variables and 100 projects with lists. Two coders classified these projects as “game” or “non-game” and reached high inter-rater reliability (Cohen’s $\kappa = 0.88$). We found that 65% (CI = [54%, 74%]) of projects with variables and 52% (CI

= [41%, 62%]) with lists were games.⁶ This reinforces our sense, developed in Study 1, that games are the dominant genre of Scratch projects containing variables and lists. It also gives us confidence in our decision to use a measure of the prevalence of games over time to test our theory related to popular genres of projects.

Because it is difficult to manually identify games in our large dataset of projects, we define projects as games if they contain the string “game” or “gaming” in their titles or descriptions. To validate this measure, we again hand-coded samples of projects from four random samples: 100 projects with variables and at least one of the strings, and 100 similar projects without the strings; two similar samples of 100 projects with lists. The same two coders coded all 400 projects as game or non-games. Among the projects that contain variables, we found that 88% (CI = [80%, 93%]) projects with strings “game” or “gaming” were games, while only 48% (CI = [38%, 58%]) projects without those strings were. We found a similar pattern among projects with lists, where 85% (CI = [76%, 91%]) and only 31% (CI = [22%, 41%]) projects were games, respectively. In other words, our method of identifying games using the strings “game” and “gaming” is high precision and somewhat low recall. Because our goal with H1 is to study change over time rather than baseline prevalence, low recall is not problematic as long as it is consistent over time. Analysis in our Appendix (§10) suggests that these proportions were consistent over the period of our study.

To test H1, we perform a logistic regression on the odds of a project with variables/lists being described as a game where the date in years in which the project was created is our independent variable. We include month-of-year fixed effects to control for seasonality. We used a linear specification of time because exploratory

⁴https://en.scratch-wiki.info/wiki/Scratch_1.3

⁵<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/2TRZ9N>

⁶ Numbers within brackets are 95% confidence intervals computed using Yates’ continuity correction.

	Variable	List
(Intercept)	-1.09*	-1.62*
	(0.02)	(0.06)
Year	-0.02*	0.06*
	(0.00)	(0.02)
Month fixed effect	yes	yes
Log Likelihood	-132558.53	-13058.12
Deviance	265117.06	26116.23
Num. obs.	241634	26440

* $p < 0.001$

Table 1: Logistic regression models for the likelihood of a project including the term “game” or “gaming” in its title or description. Models are fit on two datasets including all non-remix projects containing variables ($n = 241,634$) and all non-remix projects containing lists ($n = 26,440$) from September 2008 to April 2012.

data analysis indicated that curvilinearity was unlikely a major concern.

To test **H2** that there will be increasing concentration in variable/list names over time, we operationalize “concentration” as the Gini coefficient of the distributions of variables across names for each week of data we collected [11]. Originally invented to measure wealth inequality in a nation, Gini coefficients range from 0 representing perfect equality (if every variable name is used in an equal number of projects) to 1 reflecting perfect inequality (if only one variable name were used). We perform a linear regression using Gini Coefficient as our dependent variable and the same month-of-year fixed effects we use in H1 to control for seasonality. We use a linear specification of time for the same reason we do in H1.

H3 seeks to test the effect of exposure to popular variable/list names on subsequent behavior. We treat popular names as the 20 most frequently used names for variables or lists. Because it is not possible to measure exposure directly, we use a measure of whether a user has downloaded a project with popular variable/list names as a proxy. We feel this is justified because the only way to access the source code of a Scratch project during our data collection period was to download it. We use these measures in Cox proportional hazard models [70]. Originally developed in epidemiology, we follow the framework used by Dasgupta et al. [18] who used Cox models to measure online informal learning of computational thinking concepts in Scratch. Our models estimate the chance that a user in our dataset will share a *de novo* project with a popular variable name for the first time as a function of the number of *de novo* projects they have previously shared.

Our question predictor is a time-varying dichotomous measure of whether the user has downloaded a project with a popular variable name during our period of data collection. We conducted the same analysis for lists. Finally, we include a control variable for the total number of downloaded projects to capture overall exposure to other projects in Scratch—a potential confounder.

	Variable	List
(Intercept)	0.38*	0.11*
	(0.01)	(0.01)
Year	0.02*	0.01*
	(0.00)	(0.00)
Month fixed effect	yes	yes
R ²	0.38	0.13
Adj. R ²	0.34	0.07
Num. obs.	190	190

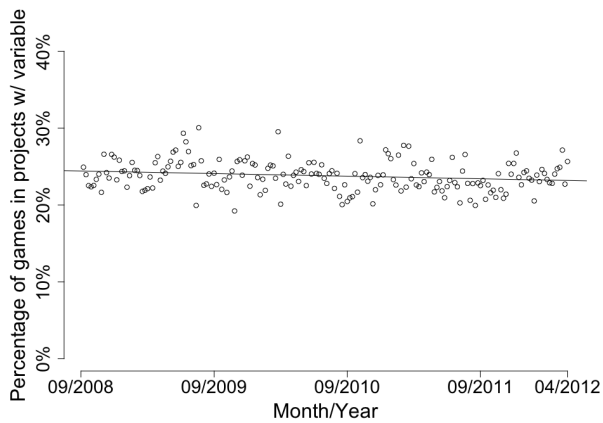
* $p < 0.001$

Table 2: OLS time series regression models on the Gini coefficient of variables across variables names for all projects shared in Scratch each week ($n = 190$).

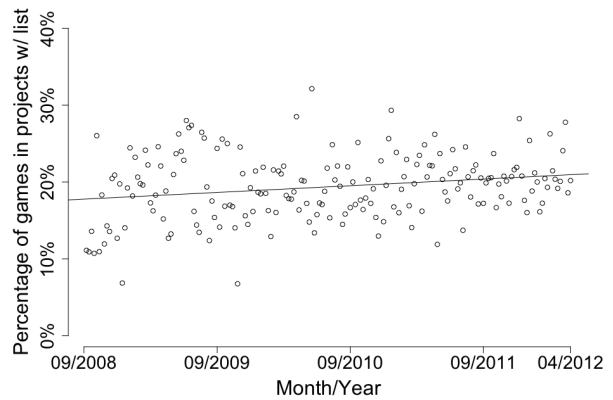
6.3 Results

The results from our hypothesis tests provide broad but uneven evidence in support of our theoretical model in §5.3. Figure 3a shows that, contrary to H1, the percentage of games in projects with variables decreased slightly over time. The hypothesis test shown in Table 1 suggests that this weak relationship is statistically significant ($\beta = -0.02$; $SE < 0.01$; $p < 0.01$) and that each year is associated with odds that are 98% the odds of the year before. On the other hand, our results for lists are in the hypothesized direction. Figure 3b shows that the percentage of games among projects with lists has been increasing over time. The results of our logistic regression in Table 1 suggest that this relationship is statistically significant ($\beta = 0.06$; $SE = 0.02$; $p < 0.01$). The model estimates that the odds that a newly created project involving a list is a game are increasing by 106% year-over-year. For instance, the model-predicted probability of a project with lists created in March 2012 being a game is 22.3%, while that of a similar project created in March 2009 is 20.8%. This estimate translates into 491 more games than we would have expected if there had been no year-over-year increase. We also checked the percentage of games among all projects (not limited to those with variables or lists) over time and found there was no obvious change in the overall game percentage. The details of this analysis is in the Appendix (§10). In other words, our findings for H1 align with our expectation on the outcome of the hypothetical social feedback loop for lists, but not for variables.

We found strong support for H2 that variable and list names would become more concentrated over time. Figure 4a shows differences in Gini coefficients over time for variables and Figure 4b shows the same measure for lists. Both figures clearly show increasing concentration. Hypothesis tests from OLS time series regression models are reported in Table 2 and reveal that these relationships are statistically significant for both variables ($\beta = 0.02$; $SE < 0.01$; $p < 0.01$) and lists ($\beta = 0.01$; $SE < 0.01$; $p < 0.01$). We estimate that the concentration across variables has increased from a Gini coefficient of about 0.41 in 2008 to 0.50 in 2012. For reference, this difference is similar to the difference in concentration of wealth between the United States (Gini coefficient = 0.41), which is more concentrated than 68% of countries globally, and Zimbabwe (Gini

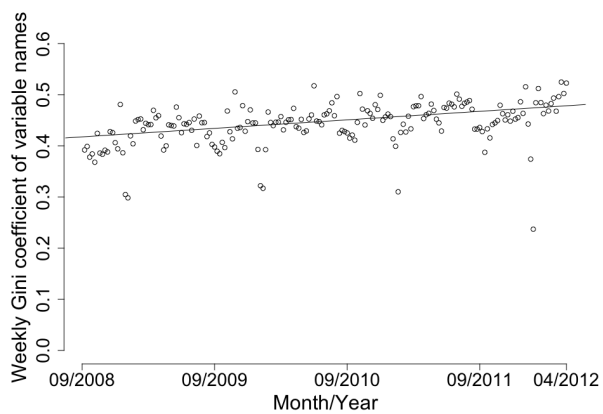


(a) Percentage of games among projects with variables.

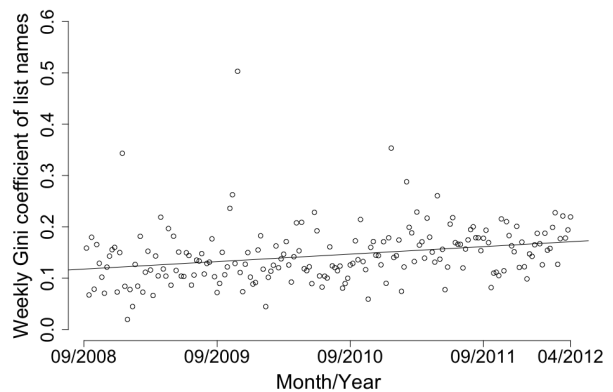


(b) Percentage of games among projects with lists.

Figure 3: Percentage of games among projects with variables or lists, per week, from September 2008 to April 2012. Lines reflect bivariate OLS regression lines.



(a) Weekly Gini coefficients of variable names.



(b) Weekly Gini coefficients of list names.

Figure 4: Weekly Gini coefficients of variable and list names over time. Lines reflect bivariate OLS regression lines.

coefficient = 0.50) which is more concentrated than 90%.⁷ In other words, the distribution of variables names is both quite concentrated and is increasing in concentration over time. Although list names are much less concentrated in general, they are increasing in concentration at a similar rate. Our findings for H2 provide additional support for what we expect to see in the community if the hypothetical social feedback loop is occurring.

Table 3 shows parameter estimates from our Cox models and shows mixed support for H3. Although we hypothesized a positive relationship between exposure to and subsequent use of popular variable names, we find that our measure of exposure to popular variables is associated with a risk of using them that is only 93% as high ($\beta = -0.07$; $SE = 0.01$; $p < 0.001$). On the other hand, users

who downloaded projects with popular list names are more likely to use those names in their own projects than those who did not. The instantaneous risk of sharing a project with a popular list name for a user who downloaded at least one project with a popular list name is 1.97 times higher than another similar user who has never downloaded such a project ($\beta = 0.68$; $SE = 0.04$; $p < 0.001$). The small negative effect associated with variables may be due to the fact that variables are used much more often than lists in Scratch so that kids may simply have more opportunities to be exposed to popular variable names.

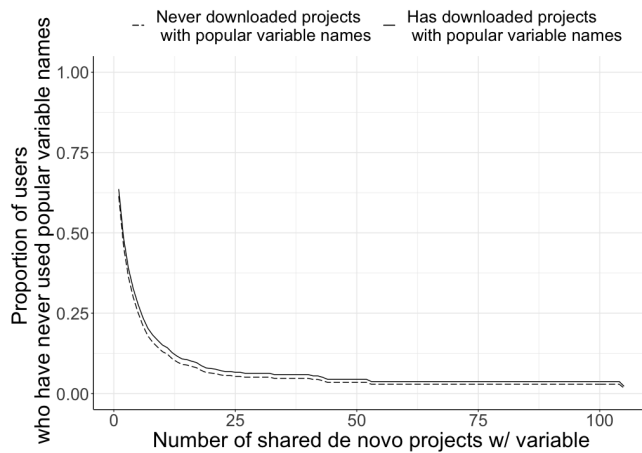
Because Cox models are difficult to interpret, we present visualizations of model-predicted estimates in Figure 5. Each panel includes two lines reflecting prototypical community members who downloaded one project before sharing projects with others: one prototypical user who had downloaded a project with a popular

⁷<https://data.worldbank.org/indicator/SI.POV.GINI>

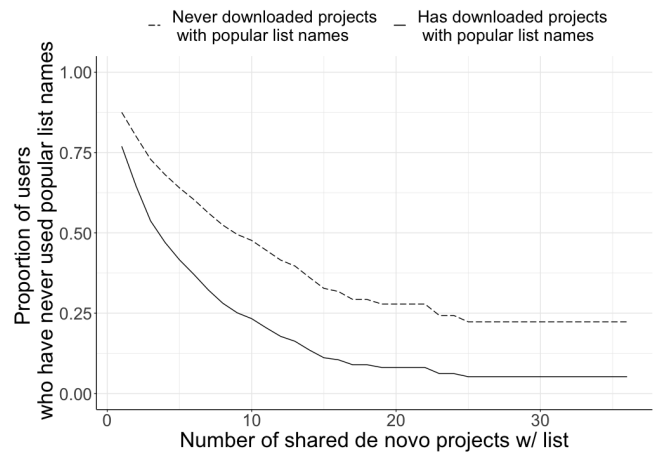
User	Risk of Using Top Variable Names	Risk of Using Top List Names
Downloaded projects w/ top variable/list names	-0.07* (0.01)	0.68* (0.04)
log(# of 100 downloads)	-0.12* (0.02)	-0.07* (0.02)
R ²	0.00	0.02
Max. R ²	1.00	0.97
Num. events	52967	3790
Num. obs.	88327	17869

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

Table 3: Fitted Cox proportional hazard models that estimate the “risk” that a Scratch user will share a *de novo* project that uses a popular variable or list name for the first time, based on whether the user has downloaded a project with top variable or list names. Number of downloads is a control to capture general exposure to other projects in Scratch. A positive coefficient means increased “risk”, while a negative coefficient means decreased “risk”.



(a) Model-predicted probability of having shared a project with a popular variable name for two prototypical users who have/have not downloaded such a project.



(b) Model-predicted probability of having shared a project with a popular list name for two prototypical users who have/have not downloaded such a project.

Figure 5: Plots of model predicted estimates of the proportion for several prototypical users. In Figure (a), estimates are shown for two prototypical users: (dashed) a user who has never downloaded projects with popular variable names, and (solid) a user who has downloaded projects with popular variable names. Figure (b) is the same plot but for lists instead of variables.

list or variable name; the other who did not. Figure 5b shows that members who have previously downloaded a project with at least one popular list name are more likely to use a popular list name in their own subsequent projects. Figure 5b shows that our model predicts that ~50% of users who have shared 10 *de novo* projects and who have never downloaded projects with popular list names will have never used a popular name in their own projects, while only ~25% of similar users who have downloaded such projects will not have. Although the negative effect is statistically significant, we do not see a similar effect with variables in Figure 5a. In summary, our findings for H3 offer some evidence for the mechanism of the social feedback loop—one again for lists but not for variables.

7 DISCUSSION: CHALLENGES & OPPORTUNITIES OF SUPPORTING LEARNING THROUGH INTEREST-DRIVEN CONTENT CREATION

While researchers have long argued that interest-driven participation can allow learners to explore and be creative [12, 46, 56, 59], our case study on computational learning in Scratch indicates that this type of participation might also create self-reinforcing social processes that constrain community imagination around advanced subjects. In our first study, we use data from the Scratch forums to build a grounded theory describing a feedback loop that exists between learners’ interests and the resources they create. This loop makes it easier for some users to learn about variables and lists—but

in ways that are increasingly focused on a set of specific functional uses that have been used by others extensively in the past. We test several hypotheses derived from this theory in a series of quantitative analyses of the Scratch code corpus and find broad, if uneven, support for the theory.

Our study contributes to the literature on computational participation by highlighting a trade-off between interest-driven participation and learning about computational concepts. On the one hand, our study shows that novice learners can learn functional uses [22] of advanced computational concepts by engaging in discussion and social support. On the other hand, we found that such learning can be superficial and not conceptual or generalizable. Echoing prior studies that raise concern about the lack of depth in the most common forms of computational participation [29, 69], our study argued that learners' preference for peer-generated learning resources around specific interests can restrict the exploration of broader and more innovative functional uses. Although it is conceivable that a narrow set of archetypal use cases could be beneficial for learning for some, increasingly homogeneous use cases stand in clear opposition to the common design goal of broadening participation.

While our empirical evidence is limited to Scratch, we speculate that our theory describes a common dynamic in informal learning environments. In the rest of this section, we discuss three challenges for Scratch and broader online learning communities implied by our theory: (1) a decrease in the diversity of resources that novice learners might draw inspiration from; (2) privileging participation by learners with the most common interests; and (3) a lack of understanding of concepts that cover broad functional use. We argue that each challenge represents promising opportunities for design.

7.1 Limited sources of inspiration

It has been argued that informal learning systems should offer “wide walls”—affordances that support a range of possibilities and pathways with which learners can creatively construct their own meanings [61, 64]. In the context of Scratch, previous research suggests that novice learners show increased engagement when the walls are “widened” through new design features [20]. Our findings describe how the unstructured nature of the Scratch online community can lead to overrepresentation of certain ways of applying knowledge—effectively “narrowing” the walls.

A range of common social features presented in Scratch and similar online artifact curation and Q&A communities are likely to reinforce this dynamic. For example, up-votes and likes may externalize the popularity of certain posts [1], artifact sharing can draw attention to already-visible topics [14], and gamified rewards can incentivize already-popular styles [10]. In each case, these features may make it difficult for learners to see beyond the limited set of popular use cases that the rest of the community is presenting. This narrowing is clearly unintended. Learners interested in a common application of a concept produce long discussion threads and an abundance of examples and tutorials out of a real desire to help. And indeed, these examples frequently *will* help others.

Inspired by the call made by Buechley and Hill [8], we suggest that future online informal learning communities should offer affordances that empower learners to leverage the “long tail” of novel use cases. Designers of the platforms should seek to help learners recognize new use cases and examples. For instance, designers might highlight novel or unusual projects and provide recognition and status to community members engaged in unconventional creation. For example, the Scratch front page has a curated section designed to showcase projects which could serve this purpose. Adaptive recommendation systems might help learners broaden their sources of inspiration by directing them to topics and genres that are different from what they are familiar with. Community moderators might guide conversations toward novel perspectives when there has been enough discussion of similar ideas.

7.2 Narrowed opportunities for participation

A related challenge is that increasingly homogeneous use cases might marginalize learners not interested in popular topics in ways that lead to demographic inequality. A number of studies in computational learning communities have shown that the underrepresentation of learners' interests and identity may give rise to a sense of being excluded or marginalized [8, 27, 65]. For instance, although many girls use Scratch, there is evidence that girls are generally less interested in making games than boys using the platform [28, 35]. As a result, game-specific learning resources related to data structures may make it easier for boys to learn about data. In that HCI researchers have built curriculum around game-making in Scratch as a way of building up computational thinking [73], we are concerned about the implications of this approach for users—disproportionately girls—who are not interested in making games.

As a possible step to address this challenge, community designers might elicit users' interests and connect users with similar interests to each other [50]. The community might also match learners with different backgrounds and interests and motivate them to exchange examples and feedback. Moderators could also offer more support and resources to users who want to explore less popular genres. For example, they might connect users with unusual interests to experts in the community. In the past, the Scratch online community has had a “welcoming committee” designed to help newcomers get started [66]. Our findings suggest a potential way to target these sorts of efforts.

7.3 Confined understanding of broader knowledge

The final challenge involves helping learners acquire an understanding of underlying computational concepts that goes beyond specific use cases. Our findings are consistent with the broader literature on learning and creativity suggesting that when a group of people engage in creative activities, they will generate less diverse ideas after having been shown popular examples [51, 81]. Our findings also echo prior work that suggests although the ability to remix can provide inspiration and scaffolds [18], there may be tradeoffs in terms of originality and whether learners might struggle to acquire transformative programming skills [33]. Our study further suggests that although community-produced examples may grow in volume over time, they may only represent material for an increasingly

narrow set of functional uses. Informal scaffolds like discussion messages and unregulated artifact catalogs may not always help learners see the big picture or master a skill.

We believe that this challenge points to a final opportunity for learning resource exploration and search systems that focus less on specific examples. For instance, hierarchical tagging and grouping mechanisms could be designed to help novice learners understand the relationship between specific examples and higher level concepts. In Scratch, a high-level collection could be called “use cases of data structures,” and the subcategories could include games, story-driven projects, and artistic media. Additionally, the discussion forum could be seeded with prompts to support the identification of underlining conceptual knowledge and to explicitly connect examples with human mentoring [26], cognitive apprenticeship [37] and automatic annotation [12].

8 LIMITATIONS

First, the interest-driven content creation investigated in this paper is limited to the setup of Scratch forums and projects. For example, Scratch users are mostly children, use block-based programming interface provided by Scratch, and tend to make programs with two-dimensional media. Other computational learning platforms or informal learning communities targeting a different subject may not include these features, and we cannot know exactly how our theory and findings will translate to other contexts. Similarly, since our theory is specifically built around variables and lists, we cannot know how it can be generalized to other type of computational learning and informal learning in general. Although in the discussion section we proposed design implications for online learning communities in general, these implications are merely speculative. Our main contribution is a case study of the Scratch community and we can know when our theories will generalize to other informal learning contexts. We share our work with the hope that future scholars will build on and critique our work by testing these theories in the communities that they study.

Scratch is used in many languages [19]. Our work is limited in that it only considers English language content. We do not know what impact the multi-lingual nature of Scratch has on our analysis or if the dynamics we observe are also present in other linguistic subcommunities in Scratch. Our strategy to detect project genre in our test of H1 is limited by language in that not all games have the words “game” or “gaming” in their title or description and some non-game projects do. Additionally, Scratch users learn from resources including project comments, tutorials, and one-to-one mentorship both within and beyond the Scratch community. In that forums are not the sole (or even primary) way that kids learn in Scratch, Study 1 might be missing important social dynamics in other places.

As we discussed in §5, our sample in Study 1 is nonrepresentative in ways that may shape our findings. Because a quarter of our sample selects on the 11 most popular variable/list names—and because these names mostly indicate game elements—our qualitative dataset may be skewed toward game-making in ways that shape our findings in Study 1. The random samples and population-level data used on Study 2 is an effort to address this issue.

Another set of limitations stems from our reliance on imperfect proxy measures in Study 2. For example, we use downloads as a

measure of exposure to test H3 because downloading was the only way to view the source code of a Scratch project before 2013. That said, users might download projects to deal with a slow internet connection or for a range of other reasons.⁸ Although we feel confident that downloads will be correlated with exposure, we have no way of knowing why a user downloaded any given project. Similarly, we used user-defined names of variables and lists as a proxy of the use cases of Scratch data structures. Although our findings from Study 1 suggest that variable and list names largely represent what users were making with variables and lists, we cannot know for sure if this is accurate in every project and there is chance that some users may use names that are inconsistent with the use case. In addition, like most other studies of informal learning online, we can only observe learning experiences and not outcomes. Measuring learning outcomes in a community like Scratch is difficult because learners arrive with different interests and aspirations and take different paths. Although we measure the presence or absence of variables and lists in projects, we can not know whether they are being used correctly or whether project creators understand the code they write [67].

Finally, although we theorize that there is a causal link between our proposed social feedback loop and increased homogenization of community-produced learning resources, we present no causal evidence. For example, our test of H3 provides evidence of a correlation between exposure to popular list names and an increased likelihood of future use of those names. This relationship might also be due to variables that are correlated with, but not caused by, a social feedback loop like the one we describe. Similarly, we try to argue that the narrowing trend in the usage of variable and list names that we discovered is an indicator of the social feedback loop that narrows the creativity in the community. However, there may be other factors outside the community, such as pop-cultural trends and school education, that contribute to this narrowing effect. To summarize, the evidence we present in Study 2 should be interpreted as similar to what we would expect to find if our theory were true—nothing more.

9 CONCLUSION

Our work contributes to HCI and social computing research by presenting a mixed-method case study about how users learn data structures through interest-driven content creation in the Scratch online community. We found evidence of a problematic, but previously untheorized, feedback loop that can constrain community innovation of functional uses. Our work describes how informal learning communities similar to Scratch could find it increasingly difficult to serve users who interests are outside the mainstream of their communities. Most importantly, our work points to several promising paths forward for designers of these systems. The problems we highlight reflect several ways that informal online learning communities could more effectively realize their incredible promise. We hope our work furthers this goal.

⁸https://en.scratch-wiki.info/wiki/Project_Downloading#Benefits_of_Downloading_Projects

ACKNOWLEDGMENTS

We would like to acknowledge members of the Community Data Science Collective—especially Emilia Gan and Stefania Druga—for their feedback and support. We also want to thank our anonymous reviewers for their feedback and support. We want to thank the Scratch team—especially Mitchel Resnick, Natalie Rusk, and Andrés Monroy-Hernandez—for building the Scratch online communities and for making data about Scratch available that allowed us to conduct this work. Finally, we owe a huge debt of gratitude to members of the Scratch online community who continue to inspire and impress us and without whom this work would not be possible.

REFERENCES

- [1] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding High-Quality Content in Social Media. In *Proceedings of the International Conference on Web Search and Web Data Mining - WSDM '08*. ACM Press, Palo Alto, California, USA, 183. <https://doi.org/10.1145/1341531.1341557>
- [2] Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286, 5439 (Oct. 1999), 509–512. <https://doi.org/10.1126/science.286.5439.509>
- [3] David Barr, John Harrison, and Leslie Conery. 2011. Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology* 38, 6 (2011), 20–23.
- [4] Paulo Blikstein. 2018. *Pre-College Computer Science Education: A Survey of the Field*. Technical Report. Google LLC, Mountain View, CA.
- [5] Karen Brennan and Mitchel Resnick. 2012. New Frameworks for Studying and Assessing the Development of Computational Thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*. AERA, Vancouver, Canada.
- [6] Karen Brennan, Amanda Valverde, Joe Prempeh, Ricarose Roque, and Michelle Chung. 2011. More than code: The significance of social interactions in young people's development as interactive media creators. In *EdMedia+ Innovate Learning*. Association for the Advancement of Computing in Education (AACE), 2147–2156.
- [7] Amy Bruckman. 1998. Community Support for Constructionist Learning. *Computer Supported Cooperative Work (CSCW)* 7, 1-2 (March 1998), 47–86. <https://doi.org/10.1023/A:1008684120893>
- [8] Leah Buechley and Benjamin Mako Hill. 2010. LilyPad in the Wild: How Hardware's Long Tail Is Supporting New Engineering and Design Communities. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems (DIS '10)*. ACM Press, New York, New York, 199–207. <https://doi.org/10.1145/1858171.1858206>
- [9] Julie Campbell, Cecilia Aragon, Katie Davis, Sarah Evans, Abigail Evans, and David Randall. 2016. Thousands of Positive Reviews: Distributed Mentoring in Online Fan Communities. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 691–704. <https://doi.org/10.1145/2818048.2819934>
- [10] Huseyin Cavusoglu, Zhuolun Li, and Ke-Wei Huang. 2015. Can Gamification Motivate Voluntary Contributions?: The Case of StackOverflow Q&A Community. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing - CSCW'15 Companion*. ACM Press, Vancouver, BC, Canada, 171–174. <https://doi.org/10.1145/2685553.2698999>
- [11] Lidia Ceriani and Paolo Verme. 2012. The Origins of the Gini Index: Extracts from *Variabilità e Mutabilità* (1912) by Corrado Gini. *The Journal of Economic Inequality* 10, 3 (Sept. 2012), 421–443. <https://doi.org/10.1007/s10888-011-9188-x>
- [12] Joel Chan, Steven Dang, and Steven P. Dow. 2016. Comparing Different Sense-making Approaches for Large-Scale Ideation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 2717–2728. <https://doi.org/10.1145/2858036.2858178>
- [13] Kathy Charmaz. 2006. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis*. Sage Publications, London, UK.
- [14] Ruijia Cheng and Mark Zachry. 2012. Building Community Knowledge In Online Competitions: Motivation, Practices and Challenges. In *Proc. ACM Hum.-Comput. Interact.* 4, CSCW2, Vol. 4. <https://doi.org/10.1145/3415250>
- [15] Ruijia Cheng, Ziwen Zeng, Maysnow Liu, and Steven Dow. 2020. Critique Me: Exploring How Creators Publicly Request Feedback in an Online Critique Community. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW2, Article 161 (Oct. 2020), 24 pages. <https://doi.org/10.1145/3415232>
- [16] Sayamindu Dasgupta. 2013. From Surveys to Collaborative Art: Enabling Children to Program with Online Data. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*. ACM, New York, NY, 28–35. <https://doi.org/10.1145/2485760.2485784>
- [17] Sayamindu Dasgupta. 2016. *Children as Data Scientists: Explorations in Creating, Thinking, and Learning*. Ph.D. Dissertation. Massachusetts Institute of Technology, Cambridge, Massachusetts.
- [18] Sayamindu Dasgupta, William Hale, Andrés Monroy-Hernández, and Benjamin Mako Hill. 2016. Remixing as a Pathway to Computational Thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, 1438–1449. <https://doi.org/10.1145/2818048.2819984>
- [19] Sayamindu Dasgupta and Benjamin Mako Hill. 2017. Learning to Code in Localized Programming Languages. In *Proceedings of the Fourth ACM Conference on Learning @ Scale (L@S '17)*. ACM, New York, NY, 33–39. <https://doi.org/10.1145/3051457.3051464>
- [20] Sayamindu Dasgupta and Benjamin Mako Hill. 2018. How “Wide Walls” Can Increase Engagement: Evidence from a Natural Experiment in Scratch. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, New York, 361:1–361:11. <https://doi.org/10.1145/3173574.3173935>
- [21] Peter J. Denning and Matti Tedre. 2019. *Computational Thinking*. MIT Press, Cambridge, MA.
- [22] Andrea A. diSessa. 1986. Models of Computation. In *User-Centered System Design: New Perspectives in Human-Computer Interaction*, Donald A. Norman and Stephen W. Draper (Eds.). Lawrence Erlbaum Associates, Hillsdale, New Jersey, 201–218.
- [23] Deborah A. Fields, Michael Giang, and Yasmin Kafai. 2014. Programming in the Wild: Trends in Youth Computational Participation in the Online Scratch Community. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*. ACM, New York, NY, USA, 2–11. <https://doi.org/10.1145/2670757.2670768>
- [24] Deborah A. Fields, Katarina Pantic, and Yasmin B. Kafai. 2015. “I Have a Tutorial for This”: The Language of Online Peer Support in the Scratch Programming Community. In *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*. ACM, New York, NY, USA, 229–238. <https://doi.org/10.1145/2771839.2771863>
- [25] Casey Fiesler, Shannon Morrison, R. Benjamin Shapiro, and Amy S. Bruckman. 2017. Growing Their Own: Legitimate Peripheral Participation for Computational Learning in an Online Fandom Community. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM, Portland Oregon USA, 1375–1386. <https://doi.org/10.1145/2998181.2998210>
- [26] Denae Ford, Kristina Lustig, Jeremy Banks, and Chris Parnin. 2018. “We Don't Do That Here”: How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, Montreal QC, Canada, 1–12. <https://doi.org/10.1145/3173574.3174182>
- [27] Denae Ford, Justin Smith, Philip J Guo, and Chris Parnin. 2016. Paradise unplugged: Identifying barriers for female participation on stack overflow. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 846–857.
- [28] Alexandra Funke and Katharina Geldreich. 2017. Gender Differences in Scratch Programs of Primary School Children. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. ACM, Nijmegen Netherlands, 57–64. <https://doi.org/10.1145/3137065.3137067>
- [29] Emilia F. Gan, Benjamin Mako Hill, and Sayamindu Dasgupta. 2018. Gender, Feedback, and Learners' Decisions to Share Their Creative Computing Projects. *Proceedings of the ACM: Human-Computer Interaction* 2, CSCW (2018), 54:1–54:23. <https://doi.org/10.1145/3274323>
- [30] James Paul Gee. 2006. *Situated Language and Learning: A Critique of Traditional Schooling* (reprinted ed.). Routledge, New York.
- [31] Colin M. Gray and Yubo Kou. 2019. Co-Producing, Curating, and Defining Design Knowledge in an Online Practitioner Community. *CoDesign* 15, 1 (Jan. 2019), 41–58. <https://doi.org/10.1080/15710882.2018.1563193>
- [32] Felienne Hermans, Alaaeddin Swidan, Efthimia Aivaloglou, and Marileen Smit. 2018. Thinking out of the Box: Comparing Metaphors for Variables in Programming Education. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education on - WiPSCE '18*. ACM Press, Potsdam, Germany, 1–8. <https://doi.org/10.1145/3265757.3265765>
- [33] Benjamin Mako Hill and Andrés Monroy-Hernández. 2013. The Remixing Dilemma The Trade-Off Between Generativity and Originality. *American Behavioral Scientist* 57, 5 (May 2013), 643–663. <https://doi.org/10.1177/0002764212469359>
- [34] Benjamin Mako Hill and Andrés Monroy-Hernández. 2017. A Longitudinal Dataset of Five Years of Public Activity in the Scratch Online Community. *Scientific Data* 4 (Jan. 2017), 170002. <https://doi.org/10.1038/sdata.2017.2>
- [35] Hui-mei Justina Hsu. 2014. Gender Differences in Scratch Game Design. In *Proceedings of the 2014 International Conference on Information, Business and Education Technology*. Atlantis Press, Beijing, China. <https://doi.org/10.2991/icibet-14.2014.28>
- [36] Z. Hui, Matthew W. Easterday, and Elizabeth M. Gerber. 2019. Distributed Apprenticeship in Online Communities. *Human-Computer Interaction* 34, 4 (July 2019), 328–378. <https://doi.org/10.1080/07370024.2018.1469409>

- [37] Julie S. Hui, Darren Gergle, and Elizabeth M. Gerber. 2018. IntroAssist: A Tool to Support Writing Introductory Help Requests. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, Montreal QC, Canada, 1–13. <https://doi.org/10.1145/3173574.3173596>
- [38] Mizuko Ito (Ed.). 2009. *Hanging Out, Messing Around, and Geeking Out: Kids Living and Learning with New Media*. The MIT Press.
- [39] Mizuko Ito, Kris Gutiérrez, Sonia Livingstone, Bill Penuel, Jean Rhodes, Katie Salen, Juliet Schor, Julian Sefton-Green, and S. Craig Watkins. 2013. *Connected Learning*. BookBaby, Cork.
- [40] Mizuko Ito, Crystle Martin, Rachel Cody Pfister, Matthew H Rafalow, Katie Salen, and Amanda Wortman. 2018. *Affinity online: How connection and shared interest fuel learning*. Vol. 2. NYU Press.
- [41] Henry Jenkins, Mizuko Itô, and danah boyd. 2016. *Participatory Culture in a Networked Era: A Conversation on Youth, Learning, Commerce, and Politics*.
- [42] Henry Jenkins, Ravi Purushotma, Margaret Weigel, Katie Clinton, and Alice J. Robison. 2009. *Confronting the Challenges of Participatory Culture: Media Education for the 21st Century*. The MIT Press, Cambridge, MA.
- [43] Joseph M Juran. 1954. Universals in Management Planning and Controlling. *Management Review* 43, 11 (1954), 748–761.
- [44] Yasmin B. Kafai. 2016. From Computational Thinking to Computational Participation in K–12 Education. *Commun. ACM* 59, 8 (July 2016), 26–27. <https://doi.org/10.1145/2955114>
- [45] Yasmin B. Kafai and Quinn Burke. 2014. *Connected Code: Why Children Need to Learn Programming*. MIT Press, Cambridge, Massachusetts.
- [46] Joy Kim, Maneesh Agrawala, and Michael S. Bernstein. 2017. Mosaic: Designing Online Creative Communities for Sharing Works-in-Progress. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*. ACM, New York, NY, USA, 246–258. <https://doi.org/10.1145/2998181.2998195> arXiv:1611.02666
- [47] Yasmine Kotturi and McKayla Kingston. 2019. Why Do Designers in the “Wild” Wait to Seek Feedback until Later in Their Design Process?. In *Proceedings of the 2019 on Creativity and Cognition*. ACM, San Diego CA USA, 541–546. <https://doi.org/10.1145/3325480.3326580>
- [48] Yubo Kou and Colin M. Gray. 2017. Supporting Distributed Critique through Interpretation and Sense-Making in an Online Creative Community. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (Dec. 2017), 1–18. <https://doi.org/10.1145/3134695>
- [49] Yubo Kou and Colin M. Gray. 2018. “What Do You Recommend a Complete Beginner like Me to Practice?”: Professional Self-Disclosure in an Online Community. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (Nov. 2018), 1–24. <https://doi.org/10.1145/3274363>
- [50] Robert E. Kraut, Paul Resnick, and Sara Kiesler. 2011. *Building Successful Online Communities: Evidence-Based Social Design*. MIT Press, Cambridge, Mass.
- [51] Chinmay Kulkarni, Steven P Dow, and Scott R Klemmer. 2014. Early and Repeated Exposure to Examples Improves Creative Work. In *Design Thinking Research*. Springer, Cham, 49–62.
- [52] Jean Lave and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, Cambridge, UK.
- [53] 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 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*. ACM Press, Vancouver, BC, Canada, 1080–1091. <https://doi.org/10.1145/2675133.2675155>
- [54] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *Trans. Comput. Educ.* 10, 4 (Nov. 2010), 16:1–16:15. <https://doi.org/10.1145/1868358.1868363>
- [55] Annette Markham. 2012. Fabrication as Ethical Practice. *Information, Communication & Society* 15, 3 (April 2012), 334–353. <https://doi.org/10.1080/1369118X.2011.641993>
- [56] Jennifer Marlow and Laura Dabbish. 2014. From Rookie to All-Star: Professional Development in a Graphic Design Social Networking Site. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '14*. ACM Press, Baltimore, Maryland, USA, 922–933. <https://doi.org/10.1145/2531602.2531651>
- [57] J. Nathan Matias, Sayamindu Dasgupta, and Benjamin Mako Hill. 2016. Skill Progression in Scratch Revisited. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM Press, New York, NY, 1486–1490. <https://doi.org/10.1145/2858036.2858349>
- [58] R. K. Merton. 1968. The Matthew Effect in Science: The Reward and Communication Systems of Science Are Considered. *Science* 159, 3810 (Jan. 1968), 56–63. <https://doi.org/10.1126/science.159.3810.56>
- [59] Andrés Monroy-Hernández. 2007. ScratchR: Sharing User-Generated Programmable Media. In *Proceedings of the 6th International Conference on Interaction Design and Children (IDC '07)*. ACM, New York, NY, 167–168. <https://doi.org/10.1145/1297277.1297315>
- [60] Seymour Papert. 1991. Situating Constructionism. In *Constructionism*, Idit Harel and Seymour Papert (Eds.). Vol. 36. Ablex Publishing, New York, NY, US, 1–11.
- [61] Seymour Papert. 1993. *Mindstorms: Children, Computers, and Powerful Ideas* (2nd ed.). Basic Books, New York.
- [62] Mitchel Resnick, Stephen Benton, Moose Crossing, and Amy Bruckman. 1998. MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids. (11 1998).
- [63] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (Nov. 2009), 60–67. <https://doi.org/10.1145/1592761.1592779>
- [64] Mitchel Resnick and Brian Silverman. 2005. Some Reflections on Designing Construction Kits for Kids. In *Proceedings of the 2005 Conference on Interaction Design and Children (IDC '05)*. ACM, New York, NY, 117–122. <https://doi.org/10.1145/1109540.1109556>
- [65] Gabriela T. Richard and Yasmin B. Kafai. 2016. Blind Spots in Youth DIY Programming: Examining Diversity in Creators, Content, and Comments within the Scratch Online Community. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) (CHI '16). Association for Computing Machinery, New York, NY, USA, 1473–1485. <https://doi.org/10.1145/2858036.2858590>
- [66] Ricarose Roque, Natalie Rusk, and Amos Blanton. 2013. Youth Roles and Leadership in an Online Creative Community. In *Proceedings of the 10th International Conference on Computer-Supported Collaborative Learning*, Vol. 1. International Society of the Learning Sciences (ISLS), Madison, WI, 399–405. <https://doi.org/10.22318/cscl2013.1.399>
- [67] Jean Salac and Diana Franklin. 2020. If They Build It, Will They Understand It? Exploring the Relationship between Student Code and Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Trondheim Norway, 473–479. <https://doi.org/10.1145/3341525.3387379>
- [68] Christopher Scaffidi, Aniket Dahotre, and Yan Zhang. 2012. How Well Do Online Forums Facilitate Discussion and Collaboration among Novice Animation Programmers?. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education - SIGCSE '12*. ACM Press, Raleigh, North Carolina, USA, 191. <https://doi.org/10.1145/2157136.2157195>
- [69] Samantha Shorey, Benjamin Mako Hill, and Samuel Woolley. 2020. From Hanging out to Figuring It out: Socializing Online as a Pathway to Computational Thinking. *New Media & Society* (May 2020), 1461444820923674. <https://doi.org/10.1177/1461444820923674>
- [70] Judith D. Singer and John B. Willett. 2003. *Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence*. Oxford University Press, Oxford ; New York.
- [71] Yla Tausczik and Ping Wang. 2017. To Share, or Not to Share?: Community-Level Collaboration in Open Innovation Contests. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (Dec. 2017), 1–23. <https://doi.org/10.1145/3134735>
- [72] Yla R. Tausczik, Aniket Kittur, and Robert E. Kraut. 2014. Collaborative Problem Solving: A Study of MathOverflow. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '14*. ACM Press, Baltimore, Maryland, USA, 355–367. <https://doi.org/10.1145/2531602.2531690>
- [73] Giovanni Maria Troiano, Qinyu Chen, Ángela Vargas Alba, Gregorio Robles, Gillian Smith, Michael Cassidy, Eli Tucker-Raymond, Gillian Puttick, and Casper Harteveld. 2020. Exploring How Game Genre in Student-Designed Games Influences Computational Thinking Development. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–17. <https://doi.org/10.1145/3313831.3376755>
- [74] Jan E. Trost. 1986. Statistically Nonrepresentative Stratified Sampling: A Sampling Technique for Qualitative Studies. *Qualitative Sociology* 9, 1 (1986), 54–57. <https://doi.org/10.1007/BF00988249>
- [75] L. S. Vygotsky. 1978. *Mind in Society*. Harvard University Press.
- [76] Jeannette M. Wing. 2006. Computational Thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. <https://doi.org/10.1145/1118178.1118215>
- [77] David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney. 2011. *App Inventor*. O'Reilly, Sebastopol, CA.
- [78] Anbang Xu and Brian Bailey. 2012. What Do You Think?: A Case Study of Benefit, Expectation, and Interaction in a Large Online Critique Community. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work - CSCW '12*. ACM Press, Seattle, Washington, USA, 295. <https://doi.org/10.1145/2145204.2145252>
- [79] Seungwon Yang, Carlotta Domeniconi, Matt Reville, Mack Sweeney, Ben U. Gelman, Chris Beckley, and Aditya Johri. 2015. Uncovering Trajectories of Informal Learning in Large Online Communities of Creators. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale (L@S '15)*. ACM, New York, NY, 131–140. <https://doi.org/10.1145/2724660.2724674>
- [80] Yu-Chun (Grace) Yen, Steven P. Dow, Elizabeth Gerber, and Brian P. Bailey. 2016. Social Network, Web Forum, or Task Market?: Comparing Different Crowd Genres for Design Feedback Exchange. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems - DIS '16*. ACM Press, Brisbane, QLD, Australia, 773–784. <https://doi.org/10.1145/2901790.2901820>

- [81] Lixiu Yu and Jeffrey V. Nickerson. 2011. Cooks or Cobblers?: Crowd Creativity through Combination. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1393–1402. <https://doi.org/10.1145/1978942.1979147>

10 APPENDIX

Figure 6 shows the trend of percentage of games among all projects (not limited to those with variables or lists) over time as part of the analysis for H1 in §6.3. As shown in the figure, the overall percentage of games stayed consistent in the time period of our study.

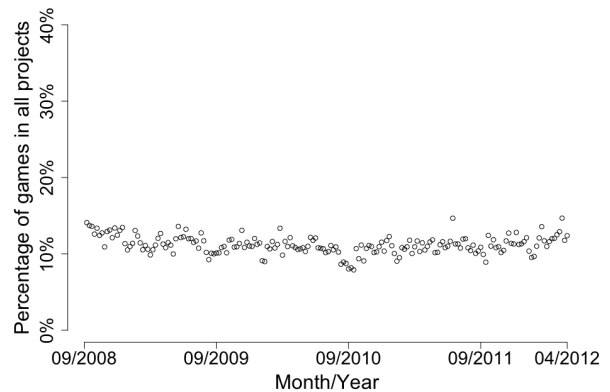


Figure 6: Percentage of games among all projects in the community, per week, from September 2008 to April 2012.

To get a sense of whether the precision and recall of our strategy to identify games stayed consistent over time, we calculated the precision and recall in the first (09/02/2008-06/22/2010) and second half (06/22/2010-04/10/2012) of our study time period, for projects with lists and projects with variables respectively. The precisions were 0.83 and 0.88 for projects with lists and 0.95 and 0.83 for projects with variables. The recalls were 0.60 and 0.62 for projects with lists and 0.56 and 0.56 for projects with variables. This means the precision and recall largely stayed consistent.