How Interest-Driven Content Creation ~~Can Limit~~ Constrains ~~Opportunities for~~ Informal Learning: A Case Study on Novices' Us~~age~~e of Data Structures in Scratch

ANONYMOUS AUTHOR(S)

Although Interest-driven content creation in online communities is cited as a promising pathway for learning, ~~but~~ it often falls short in practice. 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 ~~can increasingly spotlight certain~~ create archetyp~~al interests and thus~~es that discourage exploration ~~and diverse interests in the community~~. In a second quantitative study, we find broad support for this dynamic in several hypothesis tests. Our findings identify ~~an interest-driven~~ a social feedback loop that can restrict learning opportunities over time by limiting sources of inspiration, posing barriers to broadening participation, and confining 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**.

Additional Key Words and Phrases: 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:
Anonymous Author(s). 2018. How Interest-Driven Content Creation Can Limit Opportunities for Informal Learning: A Case Study on Novices' Usage of Data Structures in Scratch. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 51 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Scholars increasing 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 support informal learning where~~in~~ users ~~make and share artifacts by~~ engag~~ing~~e in discussion, resource sharing, and Q&A ~~that serve as~~ in ways that create learning resources for others. Although these communities exist in a range of domains like creative writing [9], graphical design [55], and more, ~~some~~ many of the largest and most studied have focused on supporting young people in learning to program. Widely ~~known~~ studied examples include the Scratch community [58] and MOOSE Crossing [61]. These communities are built around the idea that learners can develop programming skills through creating and sharing ~~coding~~ projects and interacting with other learners ~~and their creations~~ [6, 7, 44, 45].

Despite this promise, many participants in interest-driven online communities struggle ~~to become fluent in the topic that they are learning about~~. Given the opt-in and self-directed nature of participation in online communities, it is often difficult to motivate long-term engagement or to engage diverse groups of learners [14, 55, 67]. For example, in the Scratch community, steep drop-offs in engagement over time mean that most users never demonstrate a broad range of computational thinking skills or use more than a small portion of the Scratch programming language [18, 29, 56, 78]. That said, recent research suggests that increasing engagement ~~*alone*~~ is hardly a panacea to this problem. A range of studies have documented that even active ~~informal learning~~ communities face difficulties in providing ~~the requisite support for~~ learning resources ~~such as learner-generated tutorials, examples, and feedback~~ that are both high-quality and useful to a broad range of learners [1, 15, 77]. Furthermore, although discussion and Q&A in online interest-driven communities can result in exchanges of ideas, feedback, and resources, ~~many play out~~ it can also lead

~~at the level~~ of superficial socialization in ways that ~~do not dive into—and~~ can ~~even~~ act as a barrier to learning-oriented discussions [68]. ~~To this date,~~ W we still understand little about why interest driven content creation is sometimes such a rich ~~setup~~ context for informal learning while it often fall short~~, let alone how we can best design to facilitate it~~.

In this paper, we present two studies using data from the Scratch community~~. Specifically, we~~ that describe ~~a mechanism of~~ how unstructured interest-driven content creation can result in limited opportunities for informal learning~~, and identify potential ways to improve it~~. In Study 1, we present a ~~theory-building~~ grounded theory analysis of 400 discussion threads in the Scratch forums about variables and lists~~—the data structures available in Scratch~~. 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 ~~in Scratch projects~~. Through their increased visibility ~~to learners~~, these examples become archetypes ~~and go on to define or even potentially~~ that limit the breadth of future projects in the community. In Study 2, we ~~then~~ conduct a quantitative analysis of the code corpus of more than 200,000 Scratch projects ~~publicly shared between September 2008 and April 2012~~ to test our hypothesis and find statistical support for the social process theorized in our first study. ~~Our findings articulate an important trade-off in online informal learning environments in that interest-driven~~ participation ~~can generate learning resources framed in a specific set of interests while~~

~~limiting creativity and posing a barrier to broadening participation~~. We conclude with several implications for design and content curation that we believe could improve support for diverse interests.

This work makes several contributions to ~~the CHI community and~~ the HCI and social computing literature on learning ~~in online informal settings~~. First, we offer a theoretical contribution in the form of a framework describing a dynamic process where~~in~~ interest-driven content creation can both assist learning about particular topics while posing important limits on the ways that those topics are engaged with. Second, we make an empirical contribution by ~~describing the mechanism of informal learning through computational participation in online communities by~~ presenting detailed qualitative and quantitative evidence from the Scratch online community. Finally, we make a contribution to the literature on the design of informal learning systems by suggesting 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 [74] and Lave and Wenger [51]. Another key theoretical ~~pedagogical~~ framing is Papert [59]’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). ~~Papert~~ and which [60]~~’s framework~~ ~~also~~ emphasizes the importance of interest-driven exploration and “personally powerful ideas” in promoting learning. ~~The more~~ R recent scholarship ~~frameworks~~ 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 [25], 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 ~~allow~~ the products of learning are ~~to be~~ made visible as public artifacts that can be used by others as illustrative examples and for inspiration [18, 29, 55] as well as scaffolds for replication, practice, and innovation [18, 70]. To promote the second Pathway, communities feature direct user-to-user support such as comments [9], forum posts [48], and Q&A discussions [71] that can help members gain an understanding of specific topics or techniques [68]. The two pathways are deeply interwoven. By making artifacts publicly visible, creators are able to receive constructive feedback that can ~~build and~~ support learning [29, 79]. This ~~can~~ often include s input from experts and professionals that ~~are~~ is otherwise unavailable [36, 49] as well as social recognition and support [9]. Additional Learners in online communities often center their social interaction~~s~~ around discussions of public artifacts and as social interactions support the further production of artifacts [46], collaborative problem-solving [52, 68, 71] and community-wise knowledge advancement [31].

In recent years, Creative communities ~~that strive to support novices to learn~~ programming ~~skills in creative ways~~ 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," ~~a number of~~ 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 ~~projects~~ efforts inspired by this approach, millions of young people have engaged in programming in online communities ~~over~~

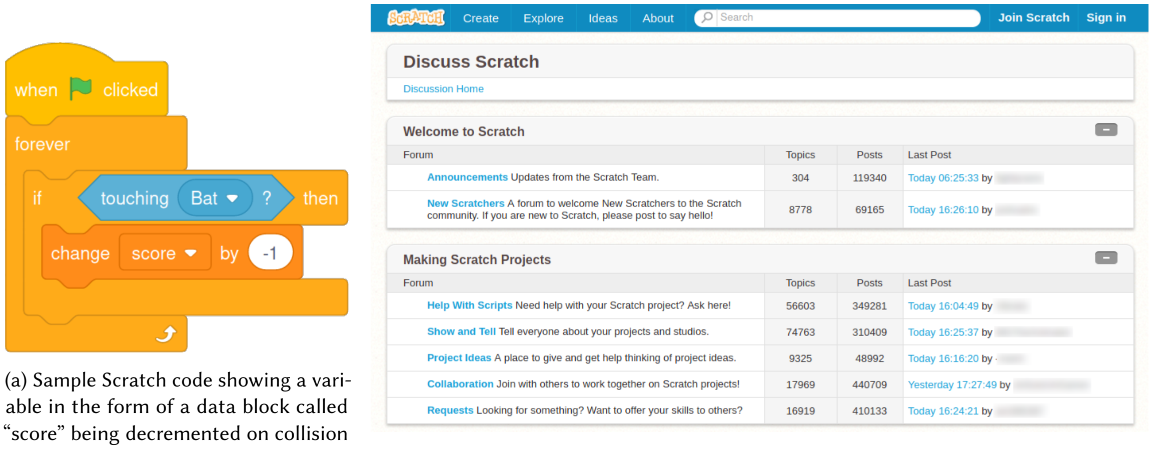
~~the last decade~~. Some vivid examples of these communities include programmable multiplayer game environments [7], platforms for interactive media creation ~~and sharing~~ [58, 62, 76], and amateur technical support groups ~~for a fanfiction website~~ [24].

Despite the promise of these communities ~~seen by many researchers and practitioners~~, it remains unclear how to best promote computational participation ~~to support learning~~. Recent studies of online programming communities have shown unequal outcomes in terms of both participation and learning based on gender [23, 29] and that important debugging or collaborative sense-making activities are not always helped by socialization [68]. Furthermore, while online communities allow users to gain inspiration from examples posted by others, studies on remixing activities suggest it might negatively impact originality [33]. These examples indicate a lack of a general understanding of the dynamics around learning ~~about computation~~ in online informal contexts. In interest-driven communities of all types, learning pathways can be blocked by the difficulty of ~~motivating and~~ ensuring ~~the~~ high quality ~~of~~ content ~~generation~~ [1, 36, 47, 55, 77] and the challenge of engaging ~~a larger and more~~ diverse groups of users ~~in creating learning resources~~ [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 ~~when~~ why user engagement supports or fails to support learning and how we can best design to facilitate ~~it~~ 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 simple st data structures comprising ~~(i.e.,~~ 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, 75]. Despite their importance, ~~learners find it challenging to learn variables and lists in interest-driven coding communities.~~ Research has shown that data is the least commonly engaged computational thinking concept in Scratch [18]~~, and even~~ W 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 variables and lists ~~a computational concept~~ involves learning their ~~its~~ structural and functional uses [22]. The *structural uses* of variables and lists (i.e., how to integrate them in a program) are straightforward~~;~~ F for example in Scratch, there are only two methods (`get()` and `set()`) that represent the structural usage of scalar variables. The *functional uses* of variables and lists (i.e., what meaningful outcome that they can help create), It ~~however,~~ are both broad and invisible. Variables can be used for ~~wide range of applications such as~~ storing user input, keeping track of internal program state, counting in a loop, and so on. Previous work has shown that Learners require~~s~~ "specific tutoring" [22, p. 207]~~,~~ or ~~effective~~ scaffolds to cope with misconceptions about how ~~a~~ variables can ~~could~~ be used ~~to make concrete functionality~~ [32]. ~~It is unclear to~~

Perhaps data use is uncommon in Scratch because



```
when 🏳 clicked
forever
  if touching Bat ? then
    change score by -1
```

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

Discuss Scratch

Discussion Home

Welcome to Scratch

Forum	Topics	Posts	Last Post
Announcements Updates from the Scratch Team.	304	119340	Today 06:25:33 by [redacted]
New Scratchers A forum to welcome New Scratchers to the Scratch community. If you are new to Scratch, please post to say hello!	8778	69165	Today 16:26:10 by [redacted]

Making Scratch Projects

Forum	Topics	Posts	Last Post
Help With Scripts Need help with your Scratch project? Ask here!	56603	349281	Today 16:04:49 by [redacted]
Show and Tell Tell everyone about your projects and studios.	74763	310409	Today 16:25:37 by [redacted]
Project Ideas A place to give and get help thinking of project ideas.	9325	48992	Today 16:16:20 by [redacted]
Collaboration Join with others to work together on Scratch projects!	17969	440709	Yesterday 17:27:49 by [redacted]
Requests Looking for something? Want to offer your skills to others?	16919	410133	Today 16:24:21 by [redacted]

(b) Index page of the Scratch forums

Fig. 1. The Scratch programming language and online community

~~what extent participation in online~~ interest-driven communities struggle to provide~~s~~ this tutoring. 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 ~~do they~~ does it fall short?* We seek to answer these questions through two closely linked empirical studies that unpack~~ing~~ practices, challenges, and opportunities for learning about variables and lists in the Scratch online community.

3 EMPIRICAL SETTING

Scratch is a visual, block-based language programming language designed for children [62]. 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~~ is described in detail by Maloney et al. [53]. When using a variable or list, Scratch users assign each variable a name by entering a user-defined string ~~to the block~~ (referred as "variable name" or "list name" in this paper). Previous work by Dasgupta et al. [18] estimates that less that 15% of Scratch users will ever make projects using data. 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 ~~through a complicated and~~ as established by an undocumented ~~process~~ 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 ~~through~~ in forums [58].[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. ~~In addition to providing toolkits for children to create projects and experiment with programming, the Scratch community supports social interactions such as appropriation~~

[1] https://scratch.mit.edu/

~~(through remixing and downloading), social connection (through "loving," "favoriting," and bookmarking projects and through following users), and discussion (through commenting and participating in discussion forums).~~ The large majority of Scratch users are between 8–16 years old and the average 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" [66].[3]

4 RESEARCH ETHICS

This research relies on two sources of ~~public~~ 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 [54] ~~and~~ to reword~~ed~~ quotes to make it more difficult to identify the Scratch users we quote using simple web searches. The first two parts of our quantitative

[2] All 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/

analysis of the Scratch code corpus in §6 relied on data that the Scratch team has published as part of the *Scratch Research Dataset* ~~and is fully reproducible~~ [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

To understand when Scratch best supports learning about the functional uses of variables and lists, we began with an open-ended interpretive analysis. This study 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 first 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 discussion, 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 [68]. The dataset that we used for sampling contains discussions from October 11, 2012, to April 5, 2017.

To acquaint ourselves with our setting, we had spent several weeks browsing the forums. As part of this process, we realized that many users struggling with problems related to variables and lists did not yet have the terminology to describe their problems as related to ‘data blocks,” “variables,” and so on. To include a broad range of conversations about data in the Scratch forums, we followed advice from Trost [73] to build a “statistically non-representative stratified sample” to ensure that a range of different ways of talking about data were reflected in our sample, but without concern for their prevalence. To do so, we sampled threads in three stages using different keywords. The distribution of the threads sampled in each stage is summarized in Table 1. First, we randomly sampled 100 threads from titles containing the keywords “variable,” “list” and “data.” Second, we sampled 10 random threads each from the 11 most common variable or list names for 110 threads total as identified in an analysis of the Scratch code corpus by Dasgupta [17]. These terms included “high 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 (half of the total sample).

Stage	Keywords	Number of threads sampled
1	"variable", "list", "data"	100
2	11 most common variable or list names	110
3	Top 100 variable or list names	200
Total threads sampled (duplicates removed): 400		

Table 1. Number of threads sampled in each stage.

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 our data Charmaz's [13] approach to constructing grounded theory. Led by the first author, we coded our data threads both line-by-line and incident-by-incident using a process of open coding. Following Charmaz, we also used a small number of "sensitizing codes" drawn from existing theories. We followed an iterative process that involved repeatedly discussing coded data as a team, conducting axial coding to group codes into themes and meta-themes, writing and discussing memos, and then returning to recode our data. 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 grounded theory analysis resulted in three major themes. First, we discovered evidence that learners, driven by specific interests in game-making, tend to adopt specific functional uses of variables and lists when making projects. 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 a trend that those specific examples can become archetypes and restrict the breadth of functional uses in the community. These findings provide support for a grounded theory about how interest-driven content creation can limit learning opportunities that we will propose in §5.3.

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 elements in the projects they were making. Because kids 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 they sought help. The concepts of variables or lists would typically be brought up by someone serendipitously in the thread. Although a quarter of the sample did not select on variable names at all and half included threads with 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 (described in §3) 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 the concept of a variable. Indeed, it is not always even obvious to users who know about 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, a kid (K1) posted in a Scratch forum: "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, users would introduce variables and their functional use as a score counter. In this case, a reply (from K2) suggested that they "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 the user, the reply highlighted the role of variables in tracking changes.

Another common pathway to learning about variables was making animated objects in games. Animation frequently relies on variables because it involves changing the speed or size of objects when triggered by certain conditions. For instance, one kid 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)

K3 arrived to the Scratch forums knowing that they wanted to vary the speed of a pong ball. For them, the challenge was not using variables *per se* but the specific case of making a ball move at a range of different speeds. Responding to their question, another kid pointed to using 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. (K4)

This response suggested that the kid asking the question use variables and told them how (as a place holder for the value of speed in the "move" operation). In these examples and many others, we saw that variables—both the concept and the term—almost never appeared in learners' initial questions. Instead, 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?" and received this answer from K6: "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 list blocks. Discussions like these played out repeatedly in the forums, helping kids who were struggling to build inventory features connect the abstract concept of a list with its functional use of storing multiple game items.

As in the variable examples above, kids imagined how an inventory would be used in the context of their games. For example, they described backpacks in which the player could select 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 including deleting and appending items and about conditions and loops. In some cases, more advanced learners would describe methods for fulfilling complicated features that a novice imagined as features of their inventory:

> 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. As with variables, these conversations typically remained focused on the specific use case of games.

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). As in the examples about variables and lists above, 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. As with lists, these conversations often segued into advanced programming topics like the encoding and decoding of strings.

In all three cases, specific use cases became linked to specific data structures—variables with counters and animation, 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 through an informal and unstructured manner. As we describe in the next section, both these conversations and the games that Scratch users created with the knowledge they built 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 situated in the context of games. These resources rarely engaged with the more general concepts behind 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 describes exactly what K13 should do to solve their problem that is specified down to the names of variables. While this answer almost certainly solved K13's problem, it did not explain either the functional or structural use of variables. Discussions like this produce learning resources that are extremely specific to the question askers' use case.

We found then 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 heartening examples of kids collaborating with and mentoring each other, the utility of the resources created is remarkably limited. The solutions typically 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. If a kid with a specific problem solvable with variables were to browse the discussion threads we analyzed, they would likely not be able to understand how variables could solve their problem unless they were making exactly the same game as someone posing a question in our sample.

Furthermore, kids offered code might be able to use it without actually understanding it [65]. Indeed, we saw kids request working solutions and others who seemed particularly happy receiving code that could be copy-and-pasted into their programs. For instance, K18 hoped the effect that they want in their game could be made into a 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? (K19)

This post was followed by a code snippet that was directly workable and had variables named as K19 imagined them.

These examples are part of a broader pattern we observed in the Scratch forums. 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 can mean that kids may not be able to see the broader picture of how variables and lists can be used. Because learning resources in communities like Scratch consist of questions and solutions accumulated over time, these examples show how community-generated learning resources can focus on specific issues in individual use cases instead of general knowledge applicable to the whole community. We explain in the next section how this high degree of specificity in knowledge resources can result in adverse learning experiences for some.

5.2.3 Specific examples become archetypes and restrict the breadth of functional uses in the community. As a result of using learning resources 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 in the broader community meant that even users without an expressed interest in games, or in some specific types of games, were often presented with resources based on them. For example, a user expressing curiosity about data blocks in very open-ended terms receives an answer that directs them to the specific functional use as a score counter in a game:

> 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 focused exclusively on them.

This reliance on canonical use cases was particularly obvious in discussions about more advanced cloud variables. For example, in the beginning of one thread, a kid first asked how cloud variables could be used without reference to any particular goal. Both responses they received assumed that the question was about making a game leaderboard:

> K22: I have no idea what cloud variable is but I heard you could make multiplayer programs 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 reinforced a set of goals and interests. Although scoring systems and leaderboards are popular in Scratch and can provide a powerful entrée to data structures, the myopic focus on these functional uses in Scratch forum would be unlikely to serve kids who have no interest making scores, leaderboards, or games.

Indeed, we found that kids with interests that deviated from canonical use cases were challenged finding learning resources that fit their interest. For instance, when K25 posed a very general question about "how to use cloud variables to save data from users," other kids tried to help them by posting examples of a high score system that involves cloud variables. In a more general thread about cloud data, 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 experience revealed what much of the Scratch forums community takes for granted. Sadly, K25 ended up not receiving help in the thread. Over time, their post was lost and ignored in a torrent of more typical messages—a large majority about leaderboards.

Unsurprisingly, we found that certain functional uses became Scratch forum's go-to examples for explaining variables and lists. It is not hard to imagine that, because learning resources were built cumulatively, more projects with these functional uses of variables and lists would be created over time. These new projects, in turn, became new learning resources for new learners. Learners who wanted to explore different use cases could do so, but 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 [60] emphasis on "personally powerful ideas" and Ito et al.'s [39] description of interest-driven, community-based learning. We found that kids are motivated to use variables and lists to solve problems and make projects they are passionate about and leverage content shared by others in the community to facilitate learning of these challenging computational concepts. Because learners making projects in Scratch are typically working with specific goals in mind, they run into the need for variables and lists while trying to implement specific functionality. They get tutored by peer-produced learning resources framed in terms of those specific functionalities. In this sense, we contribute to both 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.

Unfortunately, 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 canonical, leaving less room for unconventional interests. This leads to the possibility of a restricted set of functional uses, which can lead to shallow understanding of underlying concepts and to exclusion of learners with different goals and interests [65]. In our sample, the almost exclusive focus on certain game elements such as score counters in conversations about variables in Scratch 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 started with no specific preference and open to explore different functional uses are pointed to the canonical use cases. Overtime, this practice can make canonical examples even more archetypal, while result in user-generated content more homogeneous and less innovative.

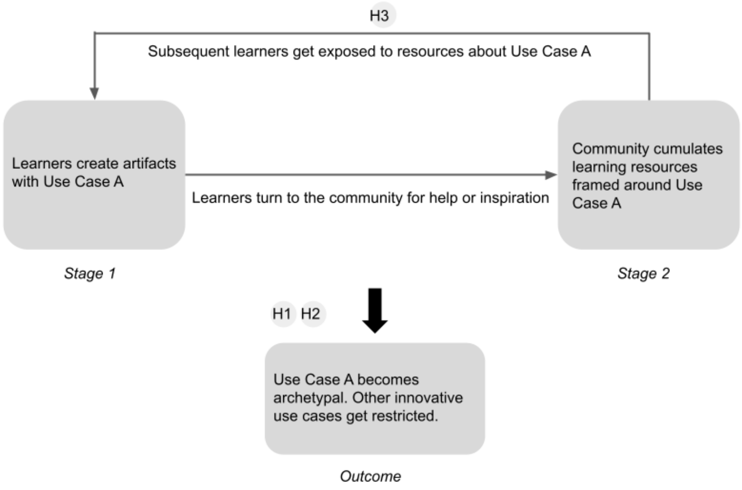

H3
Subsequent learners get exposed to resources about Use Case A
Learners create artifacts with Use Case A
Learners turn to the community for help or inspiration
Community cumulates learning resources framed around Use Case A
Stage 1
Stage 2
H1 H2
Use Case A becomes archetypal. Other innovative use cases get restricted.
Outcome

Fig. 2. Hypothesized social feedback loop in interest-driven online learning communities

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 [57], we theorize that this process plays out as a *social feedback loop*. Our social feedback loop theory is situated in the context of interest-driven online informal learning and can be summarized as follows: *interest-driven content creation can result in certain types of creation becoming archetypes, making community-generated learning resources more homogeneous and only support an increasingly limited set of learner interests over time.*

This theory is visualized in Figure 2. Stage 1 suggests that in an online interest-driven learning community, 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 user cases.

This may be due to initial user base homogeneity, targeted recruiting, 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 a archetype, the community's collective learning resources can be restricted to Use Case A. Learners like K25 in §5.2.3 who have a different interest and want to pursue new and innovative use cases receive less support. Over time, innovative use cases can become less prevalent.

6 STUDY 2: THEORY TESTING

Because the goal of our qualitative grounded theory methodology in §5 is to develop new insights, its 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 help validate the theory. Our first two hypotheses focus on the outcome of the feedback loop (marked as "H1" and "H2" on the bottom of Figure 2). The idea is that such a feedback loop will make use cases increasingly homogeneous over time and disproportionally support a small set of interests.

First, we hypothesize that certain *genres of projects* involving simple data structures will become more popular over time relative other genres. Specifically based on our findings in Study 1, we hypothesize that **(H1)** *over time, more projects involving variables and lists will be games.* As a second test, we hypothesize that certain *functional uses* of variables and lists will be increasingly archetypal. Therefore, we hypothesize that **(H2)** *the names that users give to variables and lists will become more concentrated to archetypes 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 attempts to capture part of our hypothesized mechanism. Marked as "H3" on Figure 2, we speculate that users exposed to archetypal 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.*

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 5 based on differences in the datasets we had ready 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., not remixed) 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].

6.2 Analysis and Measures

To test **H1**, we used a project-level dataset to assess whether there is an increase over time in the proportion of games with at least one variable/list in the community. To ensure that our assumption of games being a predominant genre of project was correct, we randomly

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

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. [5] This reinforces our sense developed in Study 1 that games reflect a 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 that popular genres of projects involving data will become increasingly popular. 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 inclusion criterion, we again hand coded random samples of projects as follows. We first created 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 and 85% (CI = [76%, 91%]) and only 31% (CI = [22%, 41%]) projects were games, respectively. In other words, our definition of games using the strings "game"

[5] Numbers within brackets are 95% confidence intervals computed using Yates' continuity correction.

and "gaming" would result in high precision and low recall. Because our goal with H1 is to study change over time rather than baseline prevalence, low recall is not necessarily problematic as long as it is consistent over time.

To test H1, we performed a logistic regression on the odds of a project with variables/lists being described as a game where the year in which the project was created was our independent variable. We included month-of-year fixed effects to control for seasonality. We used a linear specification of time because exploratory data analysis indicated that curvilinearity was unlikely a major concern.

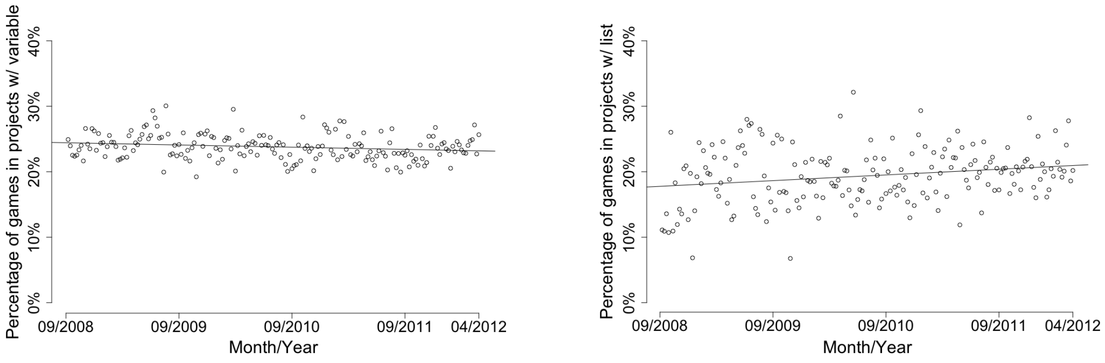
To test **H2** that there will be increasing concentration in variable/list names over time, we operationalize "concentration" as the Gini coefficient of the distributions of variables across names for each week of data we collected [11]. Originally invented to measure wealth inequality in a nation,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

H3 seeks to test the effect of exposure to popular variable/list names on subsequent behavior. We treated popular names as the 20 most frequently used names for variables or lists. Because it is not possible to measure exposure directly, we used whether a user has downloaded a project with popular variable/list names as a proxy. We do so because the only way to access the source code of a Scratch project during our data collection period was to download it. We used these measures in a survival analysis using Cox proportional hazard models [69]. Originally developed in epidemiology, we follow the framework used by Dasgupta et al. [18] who used Cox models to measure online informal learning. Our models estimate the chance that a user in our dataset will share a *de novo* (i.e., non-remix) project with a popular variable name for the first time as a function of the number of *de novo* projects they have shared.

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

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 2 suggests that this weak relationship is statistically significant ($\beta = -0.02$; SE < 0.01;



(a) Percentage of games among projects with variables.

(b) Percentage of games among projects with lists.

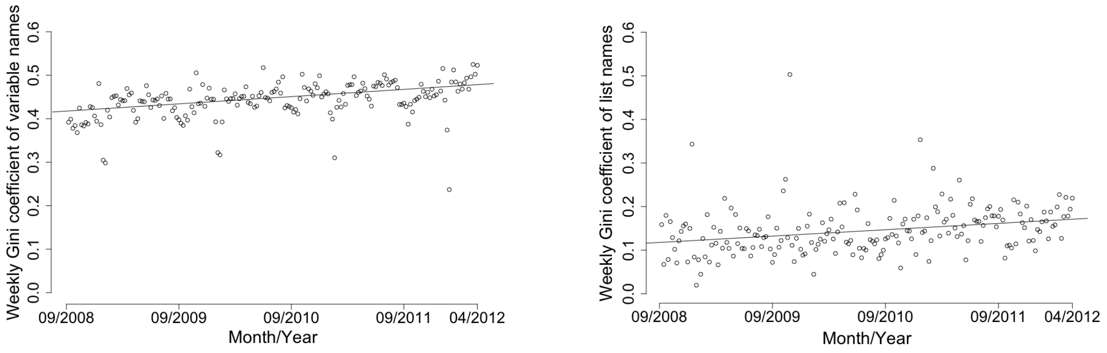
Fig. 3. Percentage of games among projects with variables or lists, per week, from September 2008 to April 2012. Lines reflect bivariate OLS regression lines.

	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
AIC	265143.06	26142.23
BIC	265278.19	26248.61
Log Likelihood	−132558.53	−13058.12
Deviance	265117.06	26116.23
Num. obs.	241634	26440

$^{*}p < 0.001$

Table 2. 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.

$p < 0.01$) 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 despite month-by-month variation. The results of our logistic regression in Table 2 suggest that this relationship is statistically significant ($\beta = 0.06$; SE $= 0.02$; $p < 0.01$).



(a) Weekly Gini coefficients of variable names.

(b) Weekly Gini coefficients of list names.

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

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

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%. There were 1,129 new projects with lists created in March 2012. This means that we estimate that there 17 more list projects that were games in that month than we would have expected if there had been no increase in concentration over the previous three years.

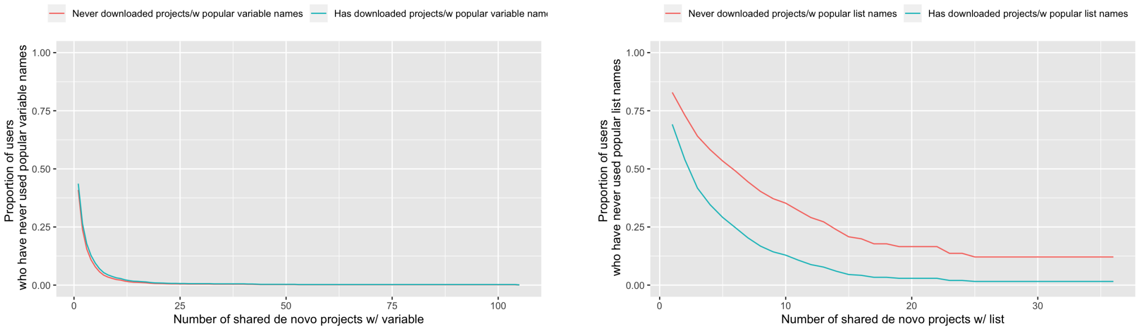
User	Risk of using popular variable names	Risk of using popular list names
Downloaded projects w/ popular variable/list names	-0.07^{*}	0.68^{*}
	(0.01)	(0.04)
log(number of 100 downloads)	-0.12^{*}	-0.07^{*}
	(0.02)	(0.02)
R^2	0.00	0.02
Num. events	52967	3790
Num. obs.	88327	17869

$^{*}p < 0.001$

Table 4. 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 popular variable or list names before. A positive coefficient means increased "risk."

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 3 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 from about 0.41 in 2008 to 0.50 in 2012. For reference, this difference is similar to the difference in concentration of wealth between United States (0.41) and Zimbabwe (0.53). [6] In other words, the distribution of variables names is quite concentrated among a few possible times and is increasing quite rapidly over time. Although list names are much less concentrated in general, they are increasing in concentration at a similar rate.

[6] https://data.worldbank.org/indicator/SI.POV.GINI



(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.

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

Table 4 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 related to 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 variable or 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 of variables may be due to the fact that variables are used much more often than lists in Scratch so that kids 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 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 sows that our model predicts that ~50% of users who have shared 5 *de novo* projects and who have never downloaded projects with popular list names would have never used a popular name in their own projects, while only ~25% of similar users who have downloaded such projects would not have. Although the negative effect effect is statistically significant, we do not see a similar effect with variables in Figure 5a.

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, 55, 58], we show how this type of participation can also create self-reinforcing social processes that lead to increasingly limited learning resources. 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 through engagement. 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 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—for example, variables and lists—by engaging in discussion and social support. On the other hand, we found that such learning is often superficial and rarely conceptual or generalizable. Echoing prior studies that concern about the lack of depth in computational participation [29, 68], our study argued that learners' preference for peer-generated learning resource around a specific and narrow set of interests can restrict 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 in some contexts, increasingly homogeneous use cases stand in clear opposition to the common design goal of broadening participation.

Although drawn from the specific data structures and project genres in Scratch, we believe that our theory describes a common dynamic in informal learning environments. In the rest of this section, we discuss three challenges for 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 a specific set of interests; and (3) a lack of understanding of concepts that cover broad functional use. We argue that these challenges represent 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 [60, 63]. 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 informal communities can lead to overrepresentation of certain ways of applying knowledge—effectively "narrowing" the walls.

This narrowing is largely unintentional, in that learners interested in a common application of a concept will produce long discussion threads and an abundance of examples and tutorials of a real desire to help. And indeed, these examples frequently *will* help others. A range of common social features in online communities might 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.

Inspired by the call made by Buechley and Hill [8], we suggest that future learning systems should offer affordances that empower learners to leverage the "long tail" of novel use cases. Designers of informal learning systems 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 use cases. 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 those topics in ways that lead to demographic inequality. A number of studies have shown that the underrepresentation of learners' interests and identity in the community may give rise to a sense of being excluded or marginalized [8, 27]. For instance, although many girls use Scratch extensively, there is evidence that girls are generally less interested in making games using the platform than boys [28, 35]. As a result, game-specific learning resources related to data structures may make it easier for boys to learn about data, on average. In that HCI researchers have built curriculum to teach game-making in Scratch as a way of building up computational thinking [72], 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. The community might also match learners with different backgrounds and interests and motivate them to exchange examples and feedback. Moderators could also deliberately 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 to make them feel welcomed. In the past, the Scratch online community has had a "welcoming committee" designed to help newcomers get started [64]. 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 concepts that goes beyond the specific use case they are interested in. Our findings are consistent with previous research suggesting that when a group of people engage in creative activities, they will generate less diverse ideas after having been shown popular examples [50]. Our findings also echo previous literature on creativity where people tend to come up with similar solutions after being exposed to others' work [50, 80]. Our work provides more evidence that informal scaffolds like discussion messages and artifacts may not always help learners see the big picture or master a skill. Our findings are consistent with prior work that suggests that although the ability to remix can provide inspiration and scaffolds [18], there may be tradeoffs in terms of the originality of remixed projects [33]. Our study suggests that although community-produced learning resources may grow in volume over time, they may represent material for an increasingly narrow set of functional uses.

We believe that this challenge points to a final opportunity for learning resource exploration and search systems that offer novice learners scaffolds 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 projects. 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

Scratch is used in many different languages and multi-lingual social content is common [19]. Our work is limited in that it both our studies only consider English language content. We do not know what impact the multi-lingual nature of Scratch has on our analysis or if the dynamics that we observe for English are also present in other linguistic subcommunities in Scratch. Our strategy to detect project genre in our test of H1 also suffers from language-related issues. For example, not all games have the words “game” or “gaming” in their title or description and some nongame projects include the term. Additionally, Scratch users learn from resources including project comments, tutorials, and one-to-one mentorship both within and beyond Scratch community. Although we do not make the argument that online discussion is the sole or primary way that kids learning in Scratch, our focus on data from the Scratch forums in Study 1 means we might be missing important social dynamics in these 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. As a result, we disclaim any attempt at establishing generalizability to any population broader than our sample in Study 1. We also address this limitation in Study 2 with random samples and population-level data. In a related sense, although our quantitative study attempted to provide tests of our theory within Scratch, we cannot know how our theory and findings will translate to other contexts. We share our work with the hope that future scholars will build on and critique our work by testing these theories in communities that they study.

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.[7] 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, and 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 come with different interests and aspirations and take different paths. Although we measure the presence or absence of data structures in projects, we can not know whether data blocks are being used correctly or whether the project creators understand them [65].

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 be due to variables that are correlated with, but not caused by, a social feedback loop like the one we describe. The evidence we present

[7] https://en.scratch-wiki.info/wiki/Project_Downloading#Benefits_of_Downloading_Projects

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 mixed method evidence of a problematic, but previously untheorized, feedback loop caused by unstructured resource creation in interest-driven informal learning environments online. Although a growing body of evidence has shown that these systems reflect increasingly important modalities for learning and that interest-driven participation can act as a powerful catalyst, these systems do not work as well as they might for every user. Our work describes how informal learning communities might find it harder 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 that goal.

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 Sensemaking 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. 2020. 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] 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

[25] Eureka Foong, Steven P. Dow, Brian P. Bailey, and Elizabeth M. Gerber. 2017. Online Feedback Exchange: A Framework for Understanding the Socio-Psychological Factors. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* ACM, Denver Colorado USA, 4454–4467. https://doi.org/10.1145/3025453.3025791

[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] Julie S. 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 Itō, 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] 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.

[51] Jean Lave and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, Cambridge, UK.

[52] 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

[53] 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

[54] Annette Markham. 2012. Fabrication as Ethical Practice. *Information, Communication & Society* 15, 3 (April 2012), 334–353. https://doi.org/10.1080/1369118X.2011.641993

[55] 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

[56] 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

[57] 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

[58] 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

[59] Seymour Papert. 1991. Situating Constructionism. In *Constructionism*, Idit Harel and Seymour Papert (Eds.). Vol. 36. Ablex Publishing, New York, NY, US, 1–11.

[60] Seymour Papert. 1993. *Mindstorms: Children, Computers, and Powerful Ideas* (2nd ed ed.). Basic Books, New York.

[61] 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).

[62] 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

[63] 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

[64] 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

[65] 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

[66] 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

[67] Abel Serrano, Javier Arroyo, and Samer Hassan. 2018. Participation Inequality in Wikis: A Temporal Analysis Using WikiChron. In *Proceedings of the 14th International Symposium on Open Collaboration*. ACM, Paris France, 1–7. https://doi.org/10.1145/3233391.3233536

[68] 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

[69] Judith D. Singer and John B. Willett. 2003. *Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence*. Oxford University Press, Oxford ; New York.

[70] 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

[71] 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

[72] 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

[73] 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

[74] L. S. Vygotsky. 1978. *Mind in Society*. Harvard University Press.

[75] Jeannette M. Wing. 2006. Computational Thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. https://doi.org/10.1145/1118178.1118215

[76] David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney. 2011. *App Inventor*. O'Reilly, Sebastopol, CA.

[77] 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

[78] Seungwon Yang, Carlotta Domeniconi, Matt Revelle, 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

[79] 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

[80] 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