Programming in the Wild: Trends in Youth Computational Participation in the Online Scratch Community

Deborah A. Fields
Utah State University
2830 Old Main Hill
Logan, UT 84321
+1 (435) 797-0571
deborah.fields@usu.edu

Michael Giang
Mount St. Mary's College
12001 Chalod Rd.
Los Angeles, CA 90049
+1 (310) 954-4157
mgiang@msmc.la.edu

Yasmin Kafai
University of Pennsylvania
3700 Walnut Street
Philadelphia, PA 19104
+1 (215) 746-3468
kafai@upenn.edu

ABSTRACT
Most research in primary and secondary computing education has focused on understanding learners within formal classroom communities, leaving aside the growing number of promising informal online programming communities where young learners contribute, comment, and collaborate on programs. In this paper, we examined trends in computational participation in Scratch, an online community with over 1 million registered youth designers primarily 11-18 years of age. Drawing on a random sample of 5,000 youth programmers and their activities over three months in early 2012, we examined the quantity of programming concepts used in projects in relation to level of participation, gender, and account age of Scratch programmers. Latent class analyses revealed four unique groups of programmers. While there was no significant link between level of online participation, ranging from low to high, and level of programming sophistication, the exception was a small group of highly engaged users who were most likely to use more complex programming concepts. Groups who only used few of the more sophisticated programming concepts, such as Booleans, variables and operators, were identified as Scratch users new to the site and girls. In the discussion we address the challenges of analyzing young learners' programming in informal online communities and opportunities for designing more equitable computational participation.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education

Keywords
Computer science education, collaborative learning, social networking sites

1. INTRODUCTION
Most efforts in primary and secondary computing education have focused on understanding learners within formal learning communities of school classrooms. Some research has examined learners' conceptions of programming—or the lack thereof [40]. Other efforts have focused on understanding novices' challenges with particular programming concepts, such as loops, conditionals, and data structures [36; 38] and focused on designing programming environments and tools that facilitate the mechanics of programming [27]. Instructional efforts have also focused on the design of programming tasks and developed approaches around more authentic and situated approaches, such as game design [21] and media-based computing [34]. Likewise social arrangements in classrooms such as pair programming [10], peer pedagogy [9] and even collaborative board games [2] have been found successful for beginning learners.

However, there is a growing interest in understanding learners in informal online communities where programming is a choice and participants contribute, comment, and collaborate on programs inspired by open source efforts [1; 19]. Unlike formal classroom communities, here participants often learn programming on their own; they program when they want, what they want, and with whom they want, with the potential to learn from and with others without the explicit guidance and support of a teacher. An early example of this type of community included MOOSE Crossing [7] while more recent examples include Kodu and Scratch. Most of the work to date has focused on Scratch (Resnick et al., 2009), by far the largest online programming community focused on youth, where young programmers post Scratch programs that they create, leave comments on each others' work, seek and provide help on forums, view, and download others' programs. To highlight the social dynamics of learning programming in these youth amateur communities, we have chosen to frame them as "computational participation" [22].

In framing interactions and contributions as computational participation, we move away from a predominantly individualistic view of computing to one that includes a greater focus on the underlying sociological and cultural dimensions in learning to code, expanding computational thinking to include social participation and personal expression. Most of the work in this area regarding youth has been primarily ethnographic in nature focusing on case studies of young designers within a large online community of programmers [3; 8]. With the recent comeback of coding and interest in promoting programming as a new literacy [22], these informal learning communities are growing in popularity (e.g., Scratch has 90,000 active monthly users as of July 2014 up from 20,000 two years ago), little is known about these learners. Our own initial efforts in studying these environments have examined levels of participation [12] and commenting [15], but nothing so far has focused on identifying

WiPSCE, November 05 - 07 2014, Berlin, Germany.

http://dx.doi.org/10.1145/2670757.2670768

the quantity and quality of programming that we can observe on such a massive scale.

In this paper, we examine youths' computational participation by choice—'in the wild'—and the type of self-organized activities that emerged with the recent arrival of online networking communities which have opened alternative paths for youth to become engaged in programming. With over seven million projects shared since its public launch in 2007, the Scratch website is a vibrant online community with 8,000-10,000 new projects being uploaded every day. Scratch is also a media-rich programming language that allows youth to design, share, and remix software programs in the form of games, stories, and animations [35]. For the purpose of our study, we drew on a random sample of 5,000 youth programmers and their activities over three months in early 2012, examining the quantity of programming concepts used in projects in relation to the level of participation, gender, and length of membership of Scratch programmers. We address the following three research questions: (1) What broad patterns exist in terms of the programming concepts kids use in the computer programs they share on the Scratch.mit.edu website? What kinds of commands do kids use in the programs they share 'in the wild'? (2) Are there any relationships between the quality of computer programs kids share and their gender or their length of membership (account lifetime) on the Scratch website? Finally, (3) Does the way someone participates in the Scratch website relate in any way to the programming content of their programs? In other words, are there any relationships between the users' programming profiles and their participation profiles? In the discussion we address the challenges of analyzing young learners' programming by choice and opportunities for designing more equitable computational participation in formal and informal online communities.

2. BACKGROUND

Our research to understand 'computational participation' of youth programming in informal communities is framed by a perspective of computing that moves away from a predominantly individualistic view of computing to one that includes a greater focus on the underlying sociological and cultural dimensions in learning to code, expanding computational thinking to include social participation and personal expression. "Computational participation… is the ability to solve problems with others, design systems for and with others, and understand the cultural and social nature of human behavior by drawing on concepts, practices, and perspectives fundamental to computer science" (p. 6, [22]). With "computational participation" we connect to recent efforts [17] to promote computational thinking that have been defined as all "aspects of designing systems, solving problems, and understanding human behaviors" that highlight the contributions of computer science (p. 6, [39]). By including this social dimension of computing, we leverage connectivity inherent in the digital world of the 21st century that becomes particularly manifest in the massive online social networking forums [16] where users post programs that they have made. In these interest-driven communities youth come together not only to hang out and mess around [2] but also to create, remix, and share their code.

Yet only a few studies have examined computational participation in these interest-driven communities, offline or online. One prime area to study computational participation at a massive level is on the Scratch website itself, as the most prominent and well-populated website where novice programmers and kids can share, comment on, and socialize around programmed Scratch projects [35]. So far, the research on computational participation in Scratch [22] has examined subsets of smaller, interest-driven communities, collabs, and individuals, in particular the prominent practice of remixing (editing and re-sharing others' projects) on the site, and also engaged in studies of interventions intended to support time-limited small-group collaboration. Most of this research has either been more observational and used case studies [5], though some studies have been more experimental in nature and have taken place within the larger online Scratch community, such as Monroy-Hernandez's broad study of remixing [31].

Our own first step in understanding broad trends of programming and participation on the Scratch website focused on participation profiles, identifying how users engaged in downloading, commenting, remixing, "loving," or friending in the online Scratch community, treating it as a kind of DIY social networking forum [12]. The examination of a random sample of 5,000 users found first that participation on the Scratch website focused on project-creation. To our surprise, creating and sharing projects was a baseline for all other kinds of online participation, demonstrating the centrality of programming and project creation on the Scratch website and providing a potentially new model for social networking sites. Further, there were almost no gender differences amongst participation profiles in terms of the different styles of participation engagement. In addition, we found that participation online shifted dramatically over the three-month time period. The majority of new users dropped their participation to lower levels, as did most users of intermediate levels of participation. Only the small group of highly involved users (high networkers, about 4% of the larger sample) had a strong likelihood of remaining in that most involved participation profile.

A follow-up study on Scratch participation focused on an examination of the quantity and quality of comments left on projects on the site [15]. We found that comments focused on the actual projects as opposed to comments more with the purpose of socializing were both more extensive and more linguistically rich and affective. This again points to the importance of project creation on the Scratch site as a center of user participation and a site of rich commenting.

In this study, we want to move beyond individual cases of success and even broad trends in online social participation to understand learners' programming in a massive informal community. The research that comes closest to the type of examination of programming on a community-wide scale was a study of the archive of Scratch programs collected on a Computer Clubhouse server over an 18 month time period [28]. In examining the scripts of over 500 Scratch programs, we found that concepts such as looping and conditionals were prominent while others such as Booleans and variables were hardly present in programs, if at all. While the archive of programs was not comprehensive (only those that clubhouse members had saved were available for analysis) it indicated what broad concepts kids struggle with and that learning programming by choice does not guarantee introduction to the wider range. With these findings as a background, in this paper we tackle the important issue of quantity and content of programming by focusing on levels of sophistication, level of participation and how gender and community membership intersects with them. We address the broad patterns of the concepts kids use in their Scratch computer programs they share on the site, possible relationships between gender and membership and the quality of computer programs kids shared, and how participation relates to the quality of programs.

3. CONTEXT

Scratch.mit.edu is a massive online community where participants, mostly youth ages 11-18 years share their computer programs [35]. Kids who share an interest in programming post animations, games, stories, science simulations, and interactive art they have made in the visual programming environment of Scratch (see Figure 1). Launched in May 2007 out of the MIT Media Lab, the Scratch site has grown to more than 2 million registered members with over 10,000 Scratch projects uploaded every day. We categorize the Scratch website as a do-it-yourself (DIY) social networking forum [16] a sub-genre of social networking site where activity centers around sharing user-created projects: projects dominate activity and social presence. User profiles are portfolio-based, showing individuals' created projects, "favorite" projects, and links to user-created galleries (collections of projects) and recent "friends" on their home page (see Figure 2). Traces of the social presence of users take the form of networking residues [16] left on projects and galleries, primarily including comments, love-its and favorites. Descriptive statistics listed under a project provide numerical values for these networking residues, listing the number of views, taggers, "love-its," favorites, remixes, downloads, and the user-curated galleries in which the project is located.

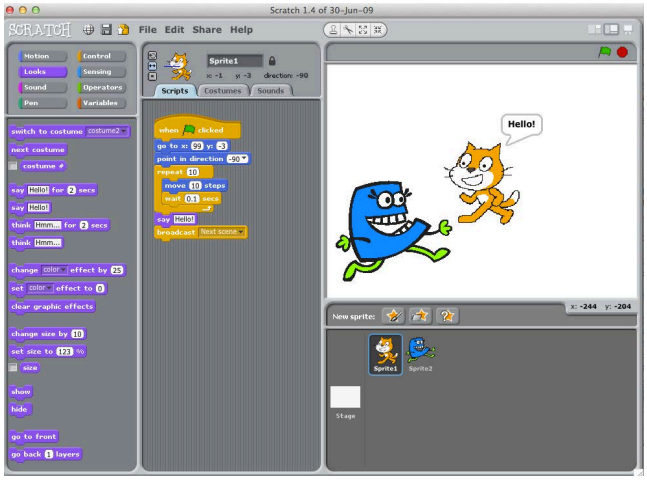

Figure 1. Scratch 1.4 programming interface.

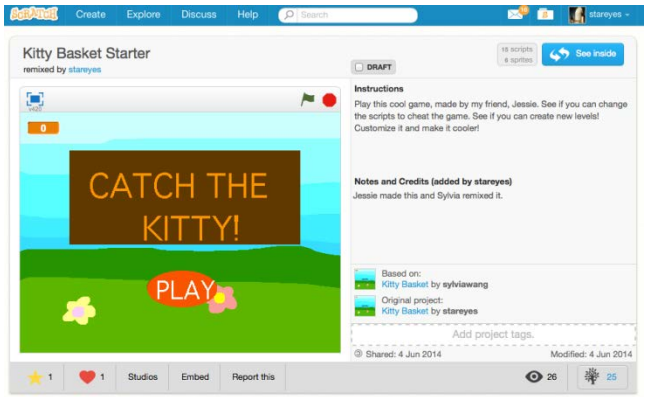

Figure 2. Scratch.mit.edu profile page showing views, love-its, favorites, and downloads.

4. METHODS

4.1 Data

Our analysis focused on a random sample of 5004 users drawn from amongst more than 20,000 users who logged into Scratch during the month of January 2012. This sample reflected the broader population on Scratch in regard to self-reported gender and age. Age on Scratch is only known through self-report (i.e., whatever birth year the user chooses). In our sample the mean age was 20 years old, the median 14, and the mode 12. However, there were a surprising number of individuals (more than 70) who were over 100 years old or under 4 years old (more than 50). Thus we view averages with great skepticism. (Similarly there are a surprising number of individuals reporting their home country as Antarctica or Aruba).

We collected data on this sample of users for three months. During these three months, 1379 users created an original project in one of the three months (January – March 2012), 533 created a project in each of two months, and 313 created a project in all three months. Thus, 2225 users (67% boys and 33% girls, reflective of the broader Scratch population) who created at least 1 project across a three-month period formed the new sample from which all further analyses reported in this paper are drawn. This sub-sample represents about 44.5% of the initial random sample of users and is the same sample used in our earlier analysis of participation profiles [12].

The reason we focus on this sub-sample is that the remaining 2779 users did no activities that we could access through the kinds of backend data available to us. In other words, though these users logged on to Scratch and likely browsed the site during the time of the study, we do not have information about what they did on Scratch. Most likely they viewed webpages without leaving any networking residues; they did not click "like" or favorite projects; they did not leave comments. Data about what users viewed was unavailable to us—it was not collected by the Scratch Team at MIT. We describe this more in our earlier analysis [12] and in section 5.4. This division of users who created and shared projects (and of those some who left comments or "love-its" or favorites) and those who did not create and share projects was a surprise to us. Based on our analyses, sharing a project on the Scratch site defines the baseline of all other active participation beyond viewing.

4.1.1 Identifying Programming Concepts in Scratch Projects

Of primary interest in this paper are the Scratch programs or *projects* users created and the kinds of programming scripts used in them. In our project data we have the number of each kind of programming block created in every new project by month. These data do not include any projects that were remixes of other projects on the Scratch site. Our logfiles contain records of the total number of *forever* loops, *broadcast* commands, and other scripts created by a user in a given month. Following the models used by Maloney and colleagues [28] and Fadjo [11] to analyze Scratch projects, we first created groupings of Scratch commands (or scripts) that fit together conceptually, including Loops, Conditionals, Broadcasts, Operators, Booleans, Sensing, and Variables (capitalized for ease of identifying these categories throughout the text). For instance, in Loops we included all non-conditional loops in Scratch, namely the *"forever"* and the *"repeat [ ] times"* loops (the only non-conditional loops in

Scratch). In Operators we included all operators that had mathematical functions[1], such as the *"[ ] + [ ]"*, *"[ ] – [ ]"*, and *"[ ] < [ ]"*. After initial analyses we determined that five categories of Scratch programming concepts helped to differentiate profile patterns amongst users: Loops, Booleans, Operators, Broadcasts, and Variables. Each of these is illustrative of the use of a particular programming concept (see also, [6]): Loops are repeating functions; Booleans involve logic statements such as *and, not,* and *or;* Operators are mathematical functions and are used only in conditional statements so they are also a stand-in for the presence of conditionals; Broadcasts are a form of synchronization and event-driven programming in Scratch; and Variables include scripts related to abstract user-created variables.

For further analysis, we divided each programming category roughly equally into four quartiles. We did this in order to see whether there were differences between creators who used larger or smaller numbers of blocks in a particular category. For instance, did it matter whether a creator used one or two loops in her programs for a month or whether she used eight or nine? Did that help to differentiate any of the users in terms of their programming? Quartile 0 (Q0) means that the user did not use any scripts involving that particular concept (i.e., a "0" in Loops means that no loops were used in that users' project within a given month). The only exception for this is Broadcast which requires at least two scripts to work (*broadcast [message]* and *when I receive [message])* so the value for Q0 is 0-1 for that concept. We divided all other values into three parts, as equivalent as possible keeping whole numbers. These are labeled 1, 2, and 3, and are shown in Table 1.

Table 1. Quartile values for programming concepts

	Loop	Boolean	Operator	Broadcast	Variable
Q0	0	0	0	0-1	0
Q1	1-3	1-2	1-6	2-4	1-7
Q2	4-11	3-10	7-22	5-11	8-27
Q3	12+	11+	23+	12+	28+

4.2 Methods

4.2.1 Programming profiles: latent class analysis of programming concepts

To identify communities of similar Scratch users based their programing preferences, we used a series of latent class analyses (LCAs) across each of three months (for more details, see [33]). The process of LCA identifies the maximum number of latent classes (groups of similar individuals) based a set of observable variables and model fit indices [18; 32]. For example, given a set of observable programming concepts (e.g., loops, booleans) and the extent they are used, LCA identifies and groups individuals with similar programming profiles into latent classes.

There are two main advantages of using LCA over other methods (e.g., mean split, cluster analyses). First, it relies on model fitting statistics and substantive interpretation to identify the optimal number of latent classes [18; 32]. LCA is an iterative process that compares models with a particular number of latent classes with a model with one less class. For example, LCA would first examine whether a model with two latent classes (e.g., novice vs. advanced users) would be provide a better fit than a one-class model (e.g., novice users). If so, LCA continues to test models with additional latent classes until model fit indices and substantive interpretation are satisfactory. Given the potential ambiguity in model fit indices, the substantive aspect of LCA allows the researcher flexibility in identifying the optimal number of latent classes to balance statistical and theoretical interpretation of each class. This avoids the potential of identifying classes with only a few users or a class that is generally similar to another except for minor statistical differences in specific observed activity. The second advantage of LCA is its ability to create latent classes with unique profiles of activities. That is, each latent class contains a profile of estimated means and corresponding probability of being classified into that class. For example, novice Scratch users may have a profile that shows an exclusive use loops at a high level, while advanced users may use loops as well as Booleans and operators at high levels. Each participant is then given probabilities of being classified into each class, and generally users have a highest-class probability (i.e., the most likely membership class).

4.2.2 Participant profiles: Membership and gender

To test whether membership or gender were proportionately represented in each of the latent classes, multiple chi-square tests for independence analyses were performed for each of the three months. These analyses were conducted based on participants' highest class assignment, self-reported gender, and membership (the total lifetime of the user's account as of January 2012). Membership was distributed across four categories of members: users with brand new accounts created in January 2012 (newbies), users with accounts up to three months old (young), accounts up to 12 months old (one-year), and accounts over one-year old (oldies). A significant chi-square test would show that there was a relationship between gender (or membership) and programming profiles. Follow-up standardize residual scores would test whether the actual count of individuals in a given cell is greater than ($z > |2|$ or $|3|$) or less than expected ($z < |2$ or $3|$) at $p = .05$ or $p = .01$. For example, a significant standardized residual would indicate that the number of females in a given membership class is significant greater ($z > 2$) or less ($z < 2$) than expected. See Table 2 for details on the distribution of membership.

Table 2. Distribution of Scratch membership

Scratch Membership	Frequency	Percent of Sample
Newbie (new account)	1436	28.7
Young (0-3 months)	1364	27.3
One-year (4-12 months)	973	19.4
Oldie (12+ months)	1165	23.3

4.2.3 Comparing relationships between programming profiles to participation profiles.

Building on our prior study of participation patterns in Scratch.mit.edu [12], we compared participation profiles with programming profiles. Our earlier study used LCA to identify the number of latent classes based on general practices of online participation. Table 3 illustrates and provides definitions for the participation profiles we identified. The study used six common participation, collaboration, and communicating practices on Scratch.mit.edu (i.e., remixing, downloading, commenting,

[1] Initially we separated mathematical operators like addition and subtraction from comparative operators such as "less than" and "equal to." However there was no difference in the users who used these commands, so we created a single category of operators.

favorites, love-its, and friend requests) to identify the number of latent classes amongst project creators. To examine whether these participation profiles were distributed equally, or rather proportionately, across the programming profiles, we conducted a chi-square test for independence analysis.

Table 3. Latent classes of Scratch users who created and shared projects in January – March 2012 (n=2225)

Name		Description
Project Creators	**Low Network**	Creates & shares projects but does nothing else visible on the site.
	Downloaders	All of the above + downloads projects
	Commenters	All of the above + comments on projects
	Networkers	All of the above + some likelihood of "love-its" or "favorites" and some friending
	High Network	All of the above + usage of both "love-its," "favorites," and friending, as well as a higher likelihood of remixing

5. FINDINGS

The results from our analyses of youth programmers in the wild fall into two broad categories: expected and unexpected findings of computational participation. On the expected side of findings, we were able to identify stable programming profiles with one class containing the most sophisticated programmers. We also found that two groups, girls and newcomers to the community, were relatively absent in that class with the most sophisticated programmers. On the unexpected side of findings, we observed an overall lack of relationship between programming profiles and participation profiles, those latter being profiles that indicate overall activity on the site as captured in the logfiles [12]. In the following sections, we explain these findings in more detail.

5.1 Programming Profiles

Our analysis identified four latent classes, or unique groups, of Scratch youth programmers, hereafter called "programming profiles" (see Appendix for more details in determining these classes). Model indices and posterior probabilities revealed very similar classes across the three months (see Figure 3). These findings indicate qualitative differences among the four groups of users who differ in their levels of programming knowledge and use; however, these classes of users are consistent across time and situations, regardless of the number of users who created projects in a given month (in January, 1770 users created projects; in February 921 users; and in March, 693 users). In describing these four unique groups, the profiles suggest different usage levels of programming concepts and thus our descriptors range from beginners to experienced (see Table 4).

Table 4. Description and percentage of user classified in each programming profiles by month (N = 1719).

Programming Profile	Description	January	February	March
Class 1: Beginners	Smaller, simpler projects: Used a relatively low percentage (quartile 1) of loops and almost no other advanced concepts.	58.4%	58.1%	51.7%
Class 2: Intermediate	Middle-sized, more complex projects: Used all programming concepts except for Booleans	18.8%	16.2%	17.3%
Class 3: Advanced	Middle-sized projects including Booleans.	19.0%	16.2%	18.0%
Class 4: Experienced	Largest projects including Booleans	8.0%	11.2%	13.0%

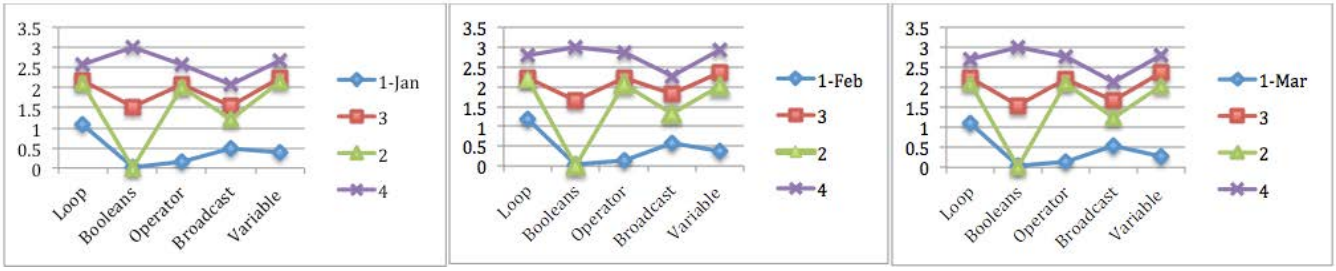

Figure 3. Latent class analyses for January, February, and March 2012, respectively

The largest class of project creators (Class 1, Beginners) uses a minimal amount of loops (quartile 1, 1-3 loops) and almost no other programming concepts we studied. These users seem to create relatively small and simple projects with few if any advanced kinds of commands. There is a relatively small likelihood that these users introduce operators, broadcasts, or variables. The second class of project creators (Class 2, Intermediate) introduces operators (i.e., conditionals) and variables into their Scratch programming projects and also utilizes more loops (quartile 2, 4-11 loops). The third and fourth classes both use Booleans. The main difference between the two is the number of Booleans and other advanced concepts they use. Class 3 (Advanced) uses roughly the same number of operators and variables as Class 2. Class 4 (Experienced) uses more of everything, suggesting to us that these projects are larger and more complex or possibly that these users simply create a larger number of projects with more (or more repetitious) advanced commands in a given month.

Our naming of these groups as Beginner, Intermediate, Advanced, and Experienced may not be reflective of the actual sophistication of these users' programs. For instance, smaller projects with advanced commands (Class 3) might actually imply greater sophistication. However, given the trends we relate later in this paper, particularly who is in the Class 4 (Experienced) group, this class appears to be more elite and more invested overall on the site.

5.2 Length of Membership Differences in Programming Profiles

Further analysis of the Scratch users' programming profiles revealed that membership, as indicated by account lifetime, matters somewhat (see Table 5). Not unexpectedly, our class of Experienced Programmers (Class 4) are less likely to be newbies (z = –2.0) and more likely to be oldtimers (z = 2.1). Confirming this trend, Intermediate Programmers (Class 2) who utilized both basic and intermediate programming are less likely to be one-year users (z = –2.7). Because these were the only observed significant differences in distributions, we conclude that length of membership does not play a large a role in terms of the content of programming. For example, within classes of Beginners and Advanced Programmers, there are (proportionally) equal number individuals across the four account lifetimes, and even within Beginners (Class 1) and Experienced (Class 4) Programmers, there are Scratch members with a year or less of being registered on the site.

Table 5. Programming profiles (class) by membership count.

	Account Lifetime			
Class	**Newbies**	**Young**	**One-year**	**Oldbies**
1	384	283	179	138
2	117	111	34~	51
3	92	66	44	42
4	37~	39	29	31+

Chi-square (9) = 24.799, p = .003

+ standardized residuals (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell.

~ standardized residuals (z) is less than -2.0; this indicates a smaller proportion of individuals than expected for the cell.

5.3 Gender Differences in Programming Profiles

In addition to length of membership in Scratch community, gender had a significant impact in which class Scratch programmers ended up. We know that girls only represent one-third of all registered members on the Scratch site (as measured by self-reported gender). The distribution in our overall sample reflected this distribution of self-reported gender on the Scratch site: 33% female and 67% male. We tested whether gender was proportionately represented in each of the latent programming classes (see Table 6). We found significant differences in regard to gender within three programming profiles, chi-square (3) = 61.359, p < .001. Specifically, there were more girls than expected in the Beginner Class (z = 3.7) and fewer girls than expected in the Advanced Class (z= –2.9) and in the Experienced Class (z= –4.1). Similarly there were fewer boys than expected in the Beginner Class (z = –2.7), and more boys than expected in the Advanced Class (z = 2.1) and especially the Experienced Class (z = 3.0).

Table 6. Programming profiles (class) by Gender count.

Class	**Boy**	**Girls**
1	575~	409++
2	217	96
3	186+	58~~
4	117++	19~~

Chi-square (3) = 61.59, p < .001

+ standardized residuals (z) is greater than 2.0; ++ z is greater than 3.3; these indicate a greater proportion of individuals than expected for the cell.

~z is less than -2.0; ~~ z is less than -3.3; these indicate a smaller proportion of individuals than expected for the cell.

5.4 Relationship between Programming Profiles and Participation Profiles

In a final step, we examined the relationship between programming profiles and participation profiles, connecting our study of programming concepts with earlier work [12] on levels of participation such as downloading, commenting, remixing, "loving," or friending in the online Scratch community. To our surprise, the analysis did not reveal a significant link, chi-square (12) = 20.791, p = .054. In general, the four classes of programmers, from beginners to experienced, existed equally across the five general participation classes and vice versa (see Table 7). Regardless of how involved users were on the Scratch site in regard to either downloading, commenting, remixing, "loving," or friending, there was no relationship to the types of programming concepts they used in their projects or how "advanced" their programming was. In other words, Scratch learners could be low networkers with intermediate or small Boolean programming profiles or commenters only programming with the simpler loops. There was, however, one exception to this trend: high networkers on the Scratch site were strongly likely to be in the Experienced Class 4 of programmers (z = 3.0). This was the *only link* between programming and participation, and again it occurred in the high pole of both programming and participation. So the most experienced Scratch users online are also the most likely to create the most advanced projects.

Table 7. Programming profiles by participation profile count.

	Programming Profiles			
Participation Profiles	**1**	**3**	**2**	**4**
Low Networker	331	68	89	33
Downloaders	223	68	85	33
Commenters	137	31	43	18
Networkers	196	46	63	25
High Networker	97	31	33	27+

Chi-square (12) = 20.791, p = .054

+ standardized residuals (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell

6. DISCUSSION

In this paper, we examined computational participation in the wild, the type of self-organized and self-directed activities that have opened alternative paths for youth to become engaged in programming. What did we learn about informal programmers? We learned that there are stable and cohesive classes of programmers in the Scratch community that reflect a range of experience based on their use of programming concepts. This work builds and expands on an earlier study of an informal programming community [28]. In the absence of other studies that have examined youth programming in informal learning communities, these findings present a first portrait of how learners in such communities are distributed. In the following sections, we discuss the nature and equity of computational participation in informal programming communities, address limitations of our analytical approach, and outline directions for future research.

6.1 Nature and Equity of Computational Participation

Our findings indicate that there is a highly experienced group of participants who are the most involved and the most advanced programmers on the Scratch site—not a surprising result given what we know from prior research in another massive online informal youth community, albeit not focused on programming [23]. While large number of participants in sites with millions of registered users result in overall high activity, it is in fact often the smallest group of users that drives the most activities. Just like in many classrooms, we have a few students who are actively participating in class activities most of the time while others are occasionally engaged and a large group of students remains at the periphery. In other words, while everyone has access to the site, not everyone is as highly engaged and contributes in the same manner in informal learning communities. It is also here where we found the most significant differences in regard to gender and membership.

We found significant differences in length of Scratch membership, or account lifetimes, as well as high networkers in regard to Experienced programming profiles. The most senior Scratch users, those who have spent the longest time on Scratch and still frequent the site, are the most likely to program with advanced programming concepts with the greatest frequency of use. On the other end of the pole, those who have created new accounts are less likely to have this programming profile. It makes some sense that those who have frequented Scratch the longest are more likely to build more complex, larger programs. Yet interestingly there are few differences between programming and participation outside of this high pole. Why might this be? One explanation is that users may join the Scratch community with diverse prior expertise in programming. Our own studies of after-school clubs have shown that some youth are reluctant to post a project to the Scratch community until they feel they have enough expertise and their projects are good enough to be comparable to others' projects [25]. Thus new users may already come with significant prior expertise in programming in Scratch. On the other hand, many classrooms and workshops have youth post projects online frequently at the end of a class for the purpose of data storage or to get feedback from others online, even if (or because) projects are incomplete and in-progress [14; 22]. This may account for a few reasons why there is such diversity of programming profile independent of participation and membership except at the highest poles of Scratch programming.

We also found significant gender differences in Beginners and Experienced classes. However, there were no significant gender differences in Intermediate Class, the group that utilized operators and variables. Proponents of broadening participation in computing may take some encouragement from this, as it suggests that at an intermediate level of novice programming in Scratch online, girls are just as well represented as boys. Further in the Advanced Class, which did use Booleans, the gender differences were much smaller than in Beginner and Experienced Classes. Our analyses indicate that the greatest differences in gender are at the extreme poles of the programming profiles.

Putting this into broader context, in our analysis of participation patterns on the Scratch website, we found almost no differences in regard to gender and participation; each profile of Scratch participation was proportionately balanced in regard to gender distribution [12]. This puts the differences in gender at the high pole of programming (Experienced) in an interesting light, especially since that class of programming showed the only significant link to a particular participation profile (the high networkers) in which there were no gender differences over time. Why is there a gender difference in programming at this high programming profile when there is no difference in gender with the linked participation profile? In the tech community, there has been a strong push to involve women in the socialization of computer science, assuming that such socialization will result in more involved and higher levels of coding. Yet these results suggest we need better understanding of how social engagement might relate to programming engagement. As Ito et al [20] might put it, hanging out and messing around, even at high levels, may not directly result in "geeking out."

Given the marked conversation on the need to broaden participation in computing [29; 30], these gender differences suggest the need not just to broaden participation in computing but also to *deepen participation* in computing, especially at the higher levels that may introduce more complex programming concepts. In other words, computational participation is not just a matter of quantity but also one of quality of engagement.

6.2 Limitations of Data and Analysis

While these findings provide us with broad trends of youth computational participation in an informal online community, they also come with certain limitations. Notably, in this dataset we cannot say *how effectively* scripts were used in Scratch projects. We can only describe *how many* of them there are in the sum total of projects uploaded by each user in a given month. Related we

cannot say exactly how large the projects are in terms of the number of programming commands. Our approximations of size in this study are solely related to the number of certain kinds of concepts used in the projects in a given month. This is admittedly a limitation in attempting to describe the sophistication and complexity of projects, however this approach was adequate given our goal of understanding of broad trends of programming in Scratch projects uploaded on a massive social networking forum.

6.3 Directions for Future Research

It is clear that we are just at the initial phase of mapping out what we know about learners in informal programming communities. Further steps should include developing better approaches to analyze programming, examining trajectories of computational participation, and broadening access to deeper computational participation. To begin with, we need address the challenges of analyzing not just the quantity but also the quality of beginners' programming. We already noted limitations in our logfile data set and how programming concepts captured different levels of engagement with Scratch. At first cut, counting the number of programming concepts in a Scratch project is a reasonable approach in capturing some level of difficulty, especially since we know from prior research that not all programming concepts are used equally in informal programming communities. We need to develop more nuanced descriptions of programming that go beyond pure project or block counts. While these first measures have been useful on a massive scale, they only reveal profiles on a surface level. Developing ways to capture context or sets of scripts on a massive scale would provide more nuanced ways to understand youth programming [13].

Furthermore, we need to situate findings from patterns in massive participation in relation to trajectories of individual learners, for bringing in individual case studies and ethnographic work on specific groups of users. We know from previous research that participants even within a class or cluster can look very different [23]. In fact we would expect quite a lot of diversity if we started looking at individuals and their trajectories of participation on the site even within a particular programming class. Examining different programming profiles, we know that programming might be easier in the beginning and then get exponentially more difficult. Moving from the Beginners to Advanced class appears to be more than just a step, it's an exponential jump. In this context, it might also be important to conduct transition analyses of kids' programming profiles over many months and possibly years. Tracing different pathways of users on the road to Advanced or Experienced programmers might provide illumination as to supports and interventions that help them reach that status. Lastly, our findings refer to an older version of the Scratch community, prior to the release of Scratch 2.0 in May 2013. The release of Scratch 2.0 could change programming profiles significantly; participation has quadrupled, users can now program directly on the site, new scripts (including functions) have been added, and many other new features incorporated such as greater ease of grabbing programming scripts from others' projects. New research is needed to see whether and how these features have facilitated different aspects of computational participation.

Another central goal of understanding learners in informal programming communities is to broaden access and deepen computational participation at large. We noted in our analysis a few significant differences in gender and membership participation that seem to replicate longstanding differences in computing communities [30]. Given the renewed interest in programming, we need to think about pathways of getting people not just into the introductory levels of programming but also into deeper, more sophisticated levels. It is a qualitative shift from broadening access to deepening participation that is not just about time spent in an activity or community but also about the kind of programming activities beginners become involved in. From some preliminary analysis on the Scratch community we know that girls become heavily involved in live role-playing games, often making thousands of comments but engaging in little programming [37]. Such projects indicate potential gateway activities into programming, but they also indicate a need for new instructional designs that foster qualitatively more complex types of programming. In this way our learning about programming in the wild can inform programming by design for broadening and deepening computational participation for all.

Finally, we need to develop better ways to connect what we know about learners and computational participation by choice—in the wild—with computational participation by design, the instructional efforts in schools. Over the last two decades, much research in K-12 education has shown the significant relationship between informal and formal learning opportunities for developing and maintaining interest as well as practicing and deepening understanding in subject matter and skills. Rather than seeing them as isolated contexts, they can be harnessed for improved outreach and for better learning in mutually beneficial ways.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by a collaborative grant from the National Science Foundation (NSF#1027736) to Mitchel Resnick, Yasmin Kafai and Yochai Benkler. The views expressed are those of the authors and do not necessarily represent the views of the National Science Foundation, Utah State University, St. Mary's College, or the University of Pennsylvania. Special thanks to Nicole Forsgren Velasquez and Taylor Martin for input on analyses and to Nicole Forsgren Velaquez, Xavier Velasquez, and Whitney King for reading earlier versions of this paper. Particular thanks to Anant Seethalakshmi for help with gathering data and to the Scratch Team for providing repeated feedback on analysis.

8. REFERENCES

[1] Y. Benkler. *The wealth of networks: How social production transforms markets and freedom*. New Haven and London: Yale University Press. 2006.

[2] M. Berland, and V. R. Lee. Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning* 1(2): 65-81, 2011.

[3] K. Brennan. Best of both worlds: Issues of structure and agency in computational creation, in and out of school. *Unpublished dissertation*, Massachusetts Institute of Technology. 2013.

[4] K. Brennan, Audience in the service of learning: How kids negotiate attention in an online community of interactive media designers. *Learning, Media, and Technology.* In press.

[5] K. Brennan and M. Resnick. Imagining, creating, playing, sharing, reflecting: How online community supports young people as designers of interactive media. In N. Lavigne and C. Mouza (Eds.), *Emerging Technologies for the Classroom: A Learning Sciences Perspective*, 5-17, 2013.

[6] K. Brennan and M. Resnick. New frameworks for studying and assessing the development of computational thinking. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada, April 2012.

[7] A. S. Bruckman. MOOSE Crossing: Construction, community, and learning in a networked virtual world for kids. Doctoral dissertation, Massachusetts Institute of Technology. 1997.

[8] A. Bruckman. Situated support for learning: Storm's weekend with Rachael. *The Journal of the Learning Sciences*, 9(3): 329-372, 2000.

[9] C. C. Ching, and Y. B. Kafai. Peer pedagogy: Student collaboration and reflection in a learning-through-design project. *The Teachers College Record*, 110(12): 2601-2632, 2008.

[10] J. Denner, L. Werner, and E. Ortiz. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, 58(1), 240-249: 2012.

[11] C. L. Fadjo. Developing computational thinking through grounded embodied cognition. *Unpublished dissertation*. Columbia University, 2012.

[12] D. A. Fields, M. Giang, and Y. B. Kafai. Understanding collaborative practices in the Scratch online community: Patterns of participation among youth designers. In N. Rummel, M. Kapur, M. Nathan, & S. Puntambekar (Eds), *To see the world and a grain of sand: Learning across levels of space, time, and scale: CSCL 2013 Conference Proceedings, Volume 1, Full Papers & Symposia*. International Society of the Learning Sciences: Madison, WI, 200-207, 2013.

[13] D. A. Fields, & H. T. Martin. *Macro data for micro learning: Developing FUN! for automated assessment of computational thinking in Scratch.* Proposal [funded]. Washington, DC: National Science Foundation grant #1319938, 2013.

[14] D. A. Fields, V. Vasudevan, and Y. B. Kafai, Y. B. The programmers' collective: Connecting collaboration and computation in a high school Scratch mashup coding workshop. In J. L. Polman, E. A. Kyza, D. K. O'Neill, I. Tabak, W. R. Penuel, A. S. Jurow, A. S., K. O'Connor, T. Lee and L. D'Amico (Eds.). *Learning and Becoming in Practice: The International Conference of the Learning Sciences (ICLS) 2014, Volume 1*. Boulder, CO: International Society of the Learning Sciences, pp. 855-862, 2014.

[15] N. Forsgren Velasquez, D. A. Fields, D. Olsen, H. T. Martin, A. Strommer, M. C. Sheperd, and Y. B. Kafai. Novice programmers talking about projects: What automated text analysis reveals about online Scratch users' comments. In the Proceedings of the *Annual Hawaii International Conference on System Sciences (HICSS)*. Waikoloa, Hawaii. IEEE, December 2013.

[16] S. M. Grimes and D. A. Fields. *Kids online: A new research agenda for understanding social networking forums.* New York. The Joan Ganz Cooney Center at Sesame Workshop. Available online at http://www.joanganzcooneycenter.org/reports-38.html. 2012.

[17] S. Grover and R. Pea. Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, 42(1): 38-43, 2013.

[18] J. Hagenaars and A. McCutcheon (Eds). *Applied Latent Class Analysis*. Cambridge, UK: Cambridge University Press, 2002.

[19] K. Healy and A. Schussman. The ecology of Open-Source software development. Working paper available at: opensource.mit.edu/papers/healyschussman.pdf (Accessed November 21, 2011), 2003.

[20] M. Ito, S. Baumer, M. Bittanti, d. boyd, R. Cody, B. Herr, H. A. Horst, P. G. Lange, D. Mahendran, K. Martinez, C. J. Pascoe, D. Perkel, L. Robinson, C. Sims and L. Tripp. *Hanging out, messing around, and geeking out: Living and learning with new media.* Cambridge, Massachusetts: MIT Press, 2010.

[21] Y. B. Kafai. *Minds in play: Computer game design as a context for children's learning*. New York, New York: Routledge, 1995.

[22] Y. B. Kafai and Q. Burke. *Connected code: Why children need to learn programming.* Cambridge, Massachusetts: MIT Press, 2014.

[23] Y. B. Kafai and D. A. Fields. Connecting play: Understanding multimodal participation in virtual worlds. In *Proceedings of the 14th ACM international conference on Multimodal interaction* (ICMI '12). ACM, New York, New York, USA, 265-272, 2012.

[24] Y. B. Kafai and D. A. Fields. *Connected Play: Tweens in a Virtual World*. Cambridge, Massachusetts: MIT Press, 2013.

[25] Y. B. Kafai, D. A. Fields and W. Q. Burke. Entering the clubhouse: Case studies of young programmers joining the online Scratch communities. *Journal of Organizational and End-User Computing*, 22(2): 21-35, 2010.

[26] Y. B. Kafai, D. A. Fields, R. Roque, W. Q. Burke and A. Monroy-Hernández . Collaborative agency in youth online and offline creative production in Scratch. *Research and Practice in Technology Enhanced Learning*, 7(2): 63-87, 2012.

[27] C. Kelleher and R. Pausch. Using storytelling to motivate programming. *Communications of the ACM,* 50(7): 58-64, 2007.

[28] J. H. Maloney, K. Peppler, Y. B. Kafai, M. Resnick and N. Rusk. Programming by choice: Urban youth learning programming with scratch. *ACM SIGCSE Bulletin,* 40(1): 367–371, 2008.

[29] J. Margolis, R. Estrella, J. Goode, J. Holme and K. Nao. *Stuck in the Shallow End: Education, Race, and Computing*. Cambridge, Massachusetts: MIT Press., 2008

[30] J. Margolis and A. Fisher. *Unlocking the Clubhouse*. Cambridge, Massachusetts: MIT Press, 2002.

[31] A. Monroy-Hernandez. Designing for remixing: Supporting an online community of amateur creators. *Unpublished dissertation*. Cambridge, Massachusetts: MIT, 2012.

[32] B. Muthen. Statistical and substantive checking in growth mixture modeling. Retrieved January 2007 from http://www.gseis.ucla.edu/faculty/muthen/full_paper_list.h, 2002.

[33] B. Muthen and L. Muthen. Integrating person-centered and variable-centered analyses: Growth mixture modeling with latent trajectory classes. Alcohol Clinical Experimental Research, 24(6): 882–891, 2001.

[34] L. Porter, M. Guzdial, C. McDowell and B. Simon. Success in introductory programming: what works? *Communications of ACM,* 56(8): 34-36, 2013.

[35] M. Resnick, J. Maloney, A. M. Hernández, N. Rusk, E. Eastmond, K. Brennan, A. D. Millner, E. Rosenbaum, J. Silver, B. Silverman and Y. B. Kafai. Scratch: Programming for everyone. *Communications of the ACM,* 52(11): 60–67, 2009.

[36] A. Robbins, J. Rountree, and N. Rountree. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education,* 13(2): 137-72, 2003.

[37] J. L. Siegel, R. Roque, D. Low, Y. B. Kafai and D. A. Fields. Understanding the creative and collaborative literacy practices in the Scratch online community: A role playing case study. Paper presented at the 33rd Annual Ethnography in Education Research Forum, Philadelphia, PA, February 2012.

[38] E. Soloway and J. Spohrer. *Empirical studies of novice programmers.* Norwood, NJ: Ablex Publishing, 1990.

[39] J. Wing. Computational Thinking. *Communications of the ACM*, 49 (3): 33-35, 2006.

[40] S. Yardi and A. Bruckman. What is computing? Bridging the gap between teenagers' perceptions and graduate students' experiences. In *Proceedings of the 3rd International Workshop on Computing Education Research.* ACM, Atlanta, GA, 39-50, 2007.

9. APPENDIX

Based on model indices (see Tables 8a-c below), LCA results identify four classes of programming users. That is, large decreases prior to and leveling off of model indices values (aBIC, BIC, AIC) after the four-class model indicate more complex models add little additional statistical depth. In addition, although LMR p-values and entropy values suggest a fifth class or higher may provide a better fit, substantive examinations of these larger classes revealed redundant classes (with marginal differences in programming use) along with small class sizes consisting of less than 1.7% of users (12-28 users among the 1719). For each participant, LCA generates probabilities for membership into each class, and generally one class has the highest probability of members. For instance, results show the January users classified into their highest probability latent class had a 95-100% probability of being in that class, and a 0-5% of being in the other classes (see Table 9a); these numbers were similar for February (96-100%; 0-4%) (Table 9b) and March (95-100%; 0-5%) (Table 9c). Taken together, the interpretation of the four-class model provided a better, more parsimonious and substantive interpretation of the data. Table 8a to 1c displays the fit indices for January, February, and March.

Table 8. Model fit indices for January to March.

Table 8a. Model fit indices for January

January: N = 1719							
# of classes	likelihood	free par	BIC	aBIC	LMR p-value	Entropy	AIC
1	-13021.72	10	26117.927	26086.158	N/A	N/A	26063.432
2	-11093.31	16	22305.81	22254.979	0.0000	0.948	22218.618
3	-10152.07	22	20468.028	20398.137	0.0000	0.95	20348.14
4	**-9699.059**	**28**	**19606.705**	**19517.752**	**0.0000**	**0.963**	**19454.119**
5	-9494.66	34	19242.602	19134.588	0.0109	0.973	19057.319
6	-8994.426	40	18286.832	18159.757	0.0112	0.999	18068.853
7	-8935.328	46	18213.332	18067.196	0.0000	0.999	17962.656

Table 8b. Model fit indices for February

February N = 1719							
# of classes	likelihood	free par	BIC	aBIC	LMR p-value	Entropy	AIC
1	-7057.718	10	14183.482	14151.724	NA	NA	14135.436
2	-5804.549	16	11717.971	11667.157	0.0000	0.967	11641.097
3	-5244.767	22	10639.236	10569.368	0.0001	0.974	10533.535
4	**-5032.713**	**28**	**10255.955**	**10167.032**	**0.0034**	**0.976**	**10121.426**
5	-4885.536	34	10002.428	9894.449	0.0002	0.983	9839.071
6	-4707.744	40	9687.673	9560.64	0.4797	0.952	9495.489
7	-4485.267	46	9283.546	9137.457	0.0072	0.999	9062.533

Table 8c. Model fit indices for March

March N = 1719							
# of classes	likelihood	free par	BIC	aBIC	LMR p-value	Entropy	AIC
1	-5355.246	10	10775.697	10743.946	NA	NA	10730.491
2	-4375.486	16	8855.302	8804.5	0.0000	0.974	8782.972
3	-4033.562	22	8210.577	8140.724	0.0011	0.966	8111.123
4	**-3902.107**	**28**	**7986.791**	**7897.888**	**0.0028**	**0.970**	**7860.214**
5	-3789.717	34	7801.134	7693.181	0.0001	0.979	7647.433
6	-3674.031	40	7608.886	7481.882	0.0080	0.997	7428.061
7	-3554.136	46	7408.221	7262.166	0.0759	0.997	7200.272

Table 9. Average Latent Class Probabilities from January to March.

Most Likely Latent Class Membership (Row) by Latent Class (Column)

Table 9a. January: Average Latent Class Probabilities

	1	**2**	**3**	**4**
1	**0.979**	0.019	0.002	0.000
2	0.001	**0.950**	0.000	0.000
3	0.050	0.000	**0.999**	0.000
4	0.000	0.000	0.000	**1.000**

Table 9b. February: Average Latent Class Probabilities

	1	**2**	**3**	**4**
1	**0.990**	0.008	0.002	0.000
2	0.005	**0.960**	0.000	0.000
3	0.040	0.002	**0.993**	0.000
4	0.000	0.000	0.000	**1.000**

Table 9c. March: Average Latent Class Probabilities

	1	**2**	**3**	**4**
1	**0.988**	0.002	0.008	0.002
2	0.003	**0.952**	0.048	0.000
3	0.000	0.011	**0.987**	0.000
4	0.000	0.000	0.000	**1.000**