Youth Computational Participation in the Wild: Understanding Experience and Equity in Participating and Programming in the Online Scratch Community

DEBORAH A. FIELDS, Utah State University
YASMIN B. KAFAI, University of Pennsylvania
MICHAEL T. GIANG, Mount Saint Mary's University

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 users contribute, comment, and collaborate on programs to facilitate learning. In this article, we examined trends in computational participation in Scratch, an online community with over 1 million registered youth designers. Drawing on a random sample of 5,004 youth programmers and their activities over 3 months in early 2012, we examined programming concepts used in projects in relation to level of participation, gender, and length of membership of Scratch programmers. Latent class analysis results identified the same four groups of programmers in each month based on the usage of different programming concepts and showed how membership in these groups shifted in different ways across time. Strikingly, the largest group of project creators (named Loops) used the simplest and fewest programming concepts. Further, this group was the most stable in membership and was disproportionately female. In contrast, the more complex programming groups (named Variables, Low Booleans, and High Booleans) showed much movement across time. Further, the Low Booleans and High Booleans groups, the only groups to use "and," "or," and "not" statements in their programs, were disproportionately male. In the discussion, we address the challenges of analyzing young learners' programming in informal online communities and opportunities for designing more equitable computational participation.

CCS Concepts: • **Social and professional topics** → **Computing education**; **Information science education**; *Computer science education*;

Additional Key Words and Phrases: Computer science education, collaborative learning, computer supported collaborative learning, social networking sites, social networking forums, educational data mining, learning analytics, equity

An earlier version of this article was published as Deborah A. Fields, Michael T. Giang, and Yasmin B. Kafai. 2014. Programming in the wild: Patterns of computational participation in the Scratch online social networking forum. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE'14)*. ACM, New York, 2–11. http://doi.acm.org/10.1145/2670757.2670768

This work is supported by the National Science Foundation (NSF #1027736).

Authors' addresses: D. A. Fields, Utah State University, 2830 Old Main Hill, Logan, UT 84322; email: deborah.fields@usu.edu; Y. B. Kafai, University of Pennsylvania, 3700 Walnut Street, Philadelphia, PA 19104; email: kafai@upenn.edu; M. T. Giang, Mount Saint Mary's University, 12001 Chalon Rd., Los Angeles, CA 90049; email: mgiang@msmu.edu.

https://doi.org/10.1145/3123815

ACM Reference format:
Deborah A. Fields, Yasmin B. Kafai, and Michael T. Giang. 2017. Youth Computational Participation in the Wild: Understanding Experience and Equity in Participating and Programming in the Online Scratch Community. *ACM Trans. Comput. Educ.* 17, 3, Article 15 (August 2017), 22 pages.
https://doi.org/10.1145/3123815

1 INTRODUCTION

Current national initiatives promote the introduction of computer science throughout K-12 education [47], especially in light of the persistent underrepresentation of women and minorities [33, 34]. Most efforts in primary and secondary computing education have focused on formal communities such as school classrooms [for overviews, see 22, 27]. An equally important charge is to understand how young learners develop in informal communities such as community technology centers [30] and online networking sites [42] where programming is a choice and participants contribute, comment, and collaborate on programs. These informal communities open alternative paths for youth to become engaged in programming, but participation is also more difficult to understand. Unlike in classroom communities, participation in informal programming activities is much less structured: there are no constraints (at least online) on when, what, with whom, or how to program, and peers (rather than formal teachers) who share a common interest in online communities can provide guidance and support [18, 19, 25].

To highlight the social dynamics around learning programming in these youth amateur communities, we have chosen to frame them as "computational participation" [27] rather than computational thinking. In framing these 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. One of the key challenges in examining computational participation in informal online communities has been how to capture the quantity and quality of programming, interactions, and connections between learners. Most of the work on Scratch, by far the largest online youth programming community [42], has been primarily ethnographic in nature, focusing on case studies of young programmers [5] or studies of smaller groups of participants [29]. While these studies lead to deep insights about sections of the Scratch community, they cannot provide a full view of the breadth of participation on the site.

Recently Berland et al. [2] proposed the use of learning analytics to study constructionist learning tools and communities. In more structured constructionist settings, learning analytics have been applied to studying a range of approaches to the same programming problem [4] or to comparing code on a specific genre of project (like video games) to set solutions [41]. By contrast, applying learning analytics in less structured constructionist communities such as Scratch presents new challenges since youth create a broad range of projects made for many different personal and social purposes. Previous studies have revealed contradictory findings of how computational concepts and practices in the Scratch community develop over time [e.g., 35, 44]. Our work aims to contribute to this growing body of research, reviewing previous approaches while also including new analytical dimensions.

In this article, we examine youths' computational participation within the Scratch community, taking into consideration membership, gender, participation patterns, and time. For the purpose of our study, we drew on a random sample of 5,004 youth programmers and their activities over three months in early 2012, examining the quantity of different programming concepts used in projects in relation to gender, length of membership, and level of participation of Scratch programmers.

We address the following three research questions: (1) Are there broad patterns of programming language use that qualitatively distinguish kids' computers programs on the scratch.mit.edu website? (2) Are there any relationships between the quality of computer programs Scratch members share and their gender, length of membership (account lifetime), and participation patterns on the website? (3) In what ways do Scratch members shift their programming patterns month by month? 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

Understanding how students learn about computational concepts such as loops, conditionals, and data structures as well as practices such as debugging and modifying program code has been a focus of research since the early 1990s [45]. Here research has examined learning and teaching of computational thinking concepts and practices to deepen understanding and broaden perspectives [21], the dynamics of social interactions in learning programming in contexts such as pair programming [9], the use of authentic programming activities such as game design [26] or media-based computing [40], and the development of programming environments and tools that facilitate the mechanics of programming [31].

But most of this research has either assessed learning of computational concepts in paper-and-pencil tasks asking individual students to answer questions to given problems or captured the development and use of computational practices in thinking-aloud protocols. Werner et al. [46] laid the groundwork for logfile data analysis of middle school students' programming in Alice, identifying "high-level actions" as a precursor to certain types of problem-solving strategies. Similarly, Berland et al. [3] used learning analytics to identify students' development of different programming strategies such as tinkering, exploring, and refinement in the course of 2 hours of programming with iPro. While such analyses move us beyond the rather static analyses of programming concepts to the more dynamic development of programming strategies, they were confined to given tasks, involved face-to-face contact, and took place in limited time frames.

Our research to understand computational participation in informal communities is driven by the need to understand the affordances and constraints that such voluntary learning contexts offer to youth. One of the first studies to examine programming on a community-wide scale was a study of the archive of Scratch programs collected at a Computer Clubhouse server over an 18-month time period [32]. The examination of scripts of over 500 Scratch programs revealed 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) and did show some improvement over time, it indicated what broad concepts clubhouse members struggled with and that learning programming by choice did not guarantee introduction to the wider range of computational concepts. This research, along with most other available studies of beginning programmers, still focused on programming activities that were conducted in a local setting requiring clubhouse members to work on their programs and interact with others in person.

With the availability of online social networking sites and programming repositories, youth can now download, share, and remix programs developed by others, thus extending the number of possible participants and projects. Scratch, as one of the most prominent websites where novice programmers and kids can share, comment on, and socialize around programmed Scratch projects [42], provides a prime area to study computational participation at a massive level. So far, the research on computational participation in Scratch has examined subsets of smaller, interest-driven communities (e.g., [1]), and also engaged in studies of interventions intended to support

time-limited small-group online collaboration between Scratch members (e.g., [16]). In particular, the prominent practice of remixing (editing and resharing others' projects) on the site has received extensive attention, establishing their relevance for creative production [36] and as a pathway into computational thinking [8].

Our own research using back-end data of the Scratch website began by focusing on participation profiles and identifying site-wide trends in how members engaged in project sharing, downloading, commenting, remixing, "loving," "favoriting," or friending in the online Scratch community. The examination of a random sample of 5,004 users found first that participation on the Scratch website focused on project creation and sharing [13], demonstrating the centrality of programming and project creation on the Scratch website. Contrary to earlier ideas, there were no significant gender differences among participation profiles. But participation online shifted dramatically over the 3-month time period, with the majority of new members dropping their participation to lower levels, as did most members of intermediate levels of participation. Only a small group of highly involved Scratch members (high networkers, about 4% of the random sample) had a strong likelihood of maintaining membership in that most involved participation profile. A follow-up study on commenting on the site found that about one-fourth of the comments on the site focused on the actual projects (as opposed to comments with a socializing purpose) and discovered these to be both extensive and more linguistically rich and affective [17]. A closer look at these comments revealed that they were largely positive and moderately constructive, generally suited to purposes of providing motivational feedback on projects and building a following of other users on the site [14].

Beyond these studies of participation, a few recent studies have returned to the idea of studying programming concepts in youths' projects. In early work, Scaffidi and Chambers [44] found contrary to their hypotheses that Scratch members with strong histories of sharing projects created less code-heavy projects and used a smaller range of blocks (i.e., fewer programming concepts) over time. A follow-up on this study by Matias et al. [35] drew from a wider swath of data—the entire population of Scratch projects from 2007 to 2012—and found that the earlier findings were due to what could be termed an "unlucky sample." Instead, they found that over time, members who stayed involved in Scratch grew in breadth and depth of programming, at least as measured by the number of programming concepts used in each project. These findings are similar to those identified by Yang et al. [48], who identified four groups of learners among very involved members on Scratch (who had posted more than 50 projects). While learning trajectories differed, all members improved in the range of blocks used in their projects. However, one limitation of all of these studies is that they focus on the trajectories of members who have already posted many projects on the Scratch site. Comparing these members to our own analysis of participation on Scratch, where only 4% of the members were likely to stay highly involved (see [10]), this is a very small sample of members overall.

Thus, we are returning to our earlier random sample of 5,004 Scratch members to tackle the important issue of quantity and content of programming by focusing on levels of program sophistication, levels of social participation, and in which ways gender and community membership intersect with them. Our sample is drawn from the back-end data of the Scratch community and contained information about the self-reported gender of Scratch members, a variable not included in previous studies [35, 44]. Furthermore, our random sample draws from the breadth of the population of Scratch, 5,004 out of 20,000 members who logged in during the month of December 2011. It includes a range of participants, newer as well as more experienced ones, and is representative of the gender distribution in the overall Scratch community at the time of sampling. These qualities will help mitigate shortcomings that Matias et al. [35] observed in previous random sampling and further address some of the challenges identified by drawing samples from the whole population.

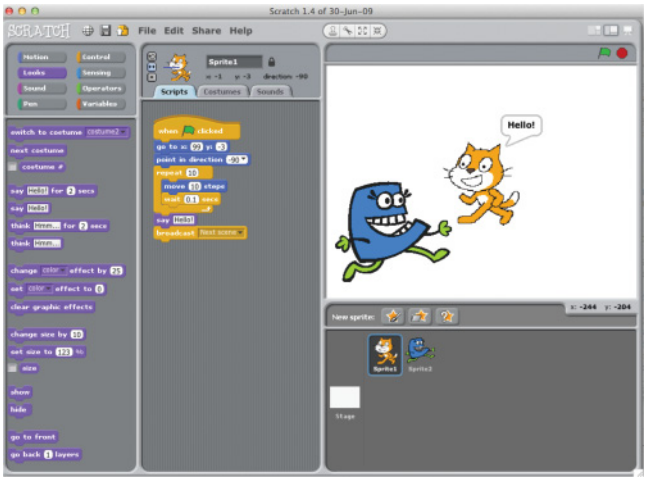

Fig. 1. Scratch 1.4 programming interface.

Examining gender, experience, and social participation on Scratch as key factors brings new variables into our analysis not previously considered in data-mining studies of youth programming communities such as Scratch. We know from related research on the online Scratch community [43] that the diversity of participants and project content is a critical issue, perhaps often overlooked with the massive number of over millions of available projects. While race is not captured in the Scratch members' profiles and available in back-end data, self-reported gender is. We address the broad patterns of the concepts Scratch members use in their computer programs they share on the site; possible relationships between gender, membership, and Scratch participation; the quality of computer programs kids shared; and how members shift between programming patterns month by month.

3 CONTEXT

The Scratch programming site (located at scratch.mit.edu) is a massive online community where participants, mostly youths ages 11 to 18 years, share their computer programs [42]. 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 16 million registered members with over 10000 Scratch projects uploaded every day. At the time of our study in early 2012, the online site was in an earlier version (Scratch 1.4), with 780,000 members and nearly 1,500 Scratch projects uploaded every day. The nature of programming on the Scratch website changed significantly with the May 2013 shift to Scratch 2.0, which allows members to program live on the site, rather than having to upload projects onto the website.

We categorize the Scratch website as a do-it-yourself (DIY) social networking forum [20], a subgenre of a social networking site where activity centers around sharing member-created projects: projects dominate activity and social presence. Member profiles are portfolio based, showing individuals' created projects, "favorite" projects, links to member-created galleries (collections of projects), and recent "friends" on their home page (see Figure 2). Traces of the social presence of members take the form of networking residues [19] left on projects and galleries, primarily including comments, "love-its," and favorites. Descriptive statistics listed under a project provide

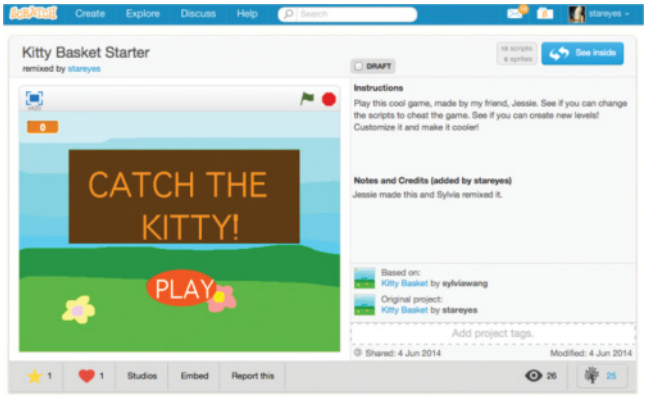

Fig. 2. Scratch.mit.edu profile page showing views, love-its, favorites, and downloads.

numerical values for these networking residues, listing the number of views, taggers, "love-its," favorites, remixes, downloads, and the member-curated galleries in which the project is located.

4 METHODS

4.1 Data

Our analysis focused on a random sample of 5,004 members drawn from among more than 20,000 members who logged into Scratch during the month of January 2012.[1] 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 member chooses at time of registration). 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.[2] (Similarly, there are a surprising number of individuals reporting their home country as Antarctica or Aruba!)

We collected data on this sample of members for 3 months. During these 3 months, 1,379 members created an original project in one of the 3 months (January—March 2012), 533 created a project in each of 2 months, and 313 created a project in all 3 months. Thus, 2,225 members (67% boys and 33% girls, reflective of the broader Scratch population) who created at least one project across a 3-month period formed the new sample from which all further analyses reported in this article are drawn. This subsample represents about 44.5% of the initial random sample of members and is the same sample used in our earlier analysis of participation profiles [13].

The reason we focus on this subsample is that the remaining 2,779 members did no activities that we could access through the kinds of back-end data available to us. In other words, though these members 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

[1]This data was gathered by the Scratch Team at MIT as part of our collaborative grant. This was facilitated by IRB permissions at both the University of Pennsylvania and MIT that allowed for aggregated data with gender and age listed. Recently the Scratch Team has released a dataset of data covering the Scratch site from 2008 to 2012 (for information and access to this dataset, see Hill and Monroy-Hernández). While much of this data overlaps with the dataset analyzed in this article, it does not include certain items that were limited by the IRB because of the intention to make the dataset broadly available by request.

[2]Because age is rather skewed and unreliable by self-report, we did not use age in our analyses in this article. Instead, we used "membership," looking at the length of time a user had been a member of the Scratch site.

leaving any networking residues; they did not click "like" or favorite projects; they did not leave comments. Data about what members viewed was unavailable to us—it was not collected by the Scratch Team at MIT. We describe these social networking activities more in our earlier analysis [13] and in Section 5.4. This division of members 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 article are the Scratch programs or *projects* members 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 used in original projects by a member in a given month. Following the models used by Maloney et al. [32], Matias et al. [35], and others (e.g., [6]) to analyze Scratch projects, we first created groupings of Scratch commands (or scripts) that fit together conceptually (see [6]), including Loops, Conditionals, Broadcasts, Operators, Booleans, Sensing, and Variables[3] (capitalized for ease of identifying these categories throughout the text). For instance, in Loops, we included all nonconditional loops in Scratch, namely, the *"forever"* and the *"repeat [ ] times"* loops (the only nonconditional loops in Scratch). In Operators, we included all operators that had mathematical functions,[4] such as the "*[ ] + [ ]*", "*[ ] − [ ]*", and "*[ ] < [ ]*".

After initial analyses, we determined five categories of Scratch programming concepts to identify profile patterns among members: Loops, Booleans, Operators, Broadcasts, and Variables. Each of these illustrates a particular programming concept (see also [7]): Loops are repeating functions; Booleans (in our study) involve the logic statements *and, not,* and *or;* Operators are mathematical functions and are used only in conditional statements in Scratch, 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 scaled the use of each programming concept into four quartiles, wherein higher scores or quartiles indicated greater use of the concept. For example, quartile 0 (Q0) means that the member did not use any scripts involving that particular concept (i.e., a "0" in Loops means that no loops were used in that member's 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 equivalently as possible, keeping whole numbers. These are labeled 1, 2, and 3 and are shown in Table 1.

Three things differentiate our data-mining approach from other analyses of the quality of programming in Scratch online community. First, we chose the concepts for our analysis based on their ability to differentiate members, rather than predeciding on a set of categories (e.g., [44]). Second, we considered not just the range of programming concepts used, but also how many times they were used, interpreted through the quartiles of programming concepts that emerged from the set of projects (e.g., [8, 35]). Third, we did not solely follow members who had a great variety of

[3]Many blocks that are familiar to current members of Scratch were not present in Scratch 1.4, the version used at the time of our study. Thus, we did not include "make your own block" commands or cloud variables as these were not present in Scratch at the time.

[4]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.

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 +

projects on Scratch (e.g., [48]); we included a random sample of all members who logged on in a given month. This allows us to speak to the full range of programming quality on Scratch at the time, rather than the range of programming quality for only the core members of Scratch.

4.2 Analysis

4.2.1 Programming Profiles: Latent Class Analysis of Programming Concepts. To identify communities of similar Scratch members based their programing preferences, we conducted a series of latent class analyses (LCAs) across each of 3 months [38]. The process of LCA identifies the maximum number of latent classes (groups of similar individuals) based on a set of observable variables and model fit indices [23, 37]. 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 [23, 37]. 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 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 members or a class that is generally similar to another except for minor statistical differences in a 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 members may have a profile that shows an exclusive use of loops at a high level, while advanced users may use loops as well as Booleans and operators at high levels. Each member is then given probabilities of being classified into each class; however, as is the case in our study, users are then categorized into their highest-probability class (i.e., the most likely membership class) for additional analyses.

4.2.2 Participant Profiles: Membership and Gender. To test whether length of 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 3 months. These analyses were conducted based on members' 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: members with brand-new accounts created in January 2012 (newbies), members with accounts up to 3 months old (young), accounts up to 12 months old (1 year), and accounts over 1 year old (oldies). A significant chi-square test would show that there was a relationship between

Table 2. Distribution of Scratch Membership

Scratch Membership	Frequency	Percent of Sample
Newbie (new account)	1,436	28.7
Young (0–3 months)	1,364	27.3
One-year (4–12 months)	973	19.4
Oldie (12 + months)	1,165	23.3

gender (or membership) and programming profiles. Follow-up standardized residual scores (Pearson) would test whether the actual count of individuals in a given cell is proportionally greater than ($z > |2|$) or less than expected ($z < |2|$) at $p < .05$).[5] For example, a significant standardized residual would indicate that the number of females in a given membership class is significantly greater ($z > 2$) or less ($z < -2$) than expected. See Table 2 for details on the distribution of membership.

4.2.3 Changing Membership in Programming Profiles by Month. We investigated changing membership between programming classes in two related ways. First, using the LCA results, members were classified into qualitatively different programming profiles based on their highest-probability class for each of the 3 months. Each profile described the extent of programming concept usage across multiple codes. These programming profiles were then compared to examine how participants' programming connected across time. Because membership in these groups requires Scratch members to create content each of the 3 months, the number of members in these analyses change by month (January N = 1,719 February N = 902, March N = 679). Second, as LCA was used to classify members into specific programming profiles (or latent classes), the individual probabilities, which can be interpreted as the likelihood of using each programming pattern, will also be used to link the usage among the different programming patterns across time.

5 FINDINGS

The results from our analyses of youth programmers in the wild reveal some interesting trends of computational participation in Scratch at the time of the study. We were able to identify four stable programming profiles, including two classes that used all of the included programming concepts in the study, one of which used larger quantities of these concepts (Class 4 or High Booleans). We also found that two groups, girls and newcomers to the community, were relatively absent in the class that used the highest levels of all the programming concepts studied (i.e., Class 4). Looking at how Scratch members shifted between programming profiles, we found that one class tended to have more enduring membership: the most elementary class of programmers (i.e., Class 1 or Loops). The three classes of Scratch members who were using what many educators would consider to be more challenging programming concepts tended to shift among themselves, revealing some fluidity in the kinds of programming their projects contained. In the following sections, we explain these findings and trends 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 profiles). Model indices and posterior probabilities revealed very similar classes across the 3 months (see Figure 3). These findings indicate qualitative differences among the four profiles of Scratch members who differ in their levels of programming knowledge and use; however, these classes of

[5]Haberman-adjusted residuals also yielded similar results.

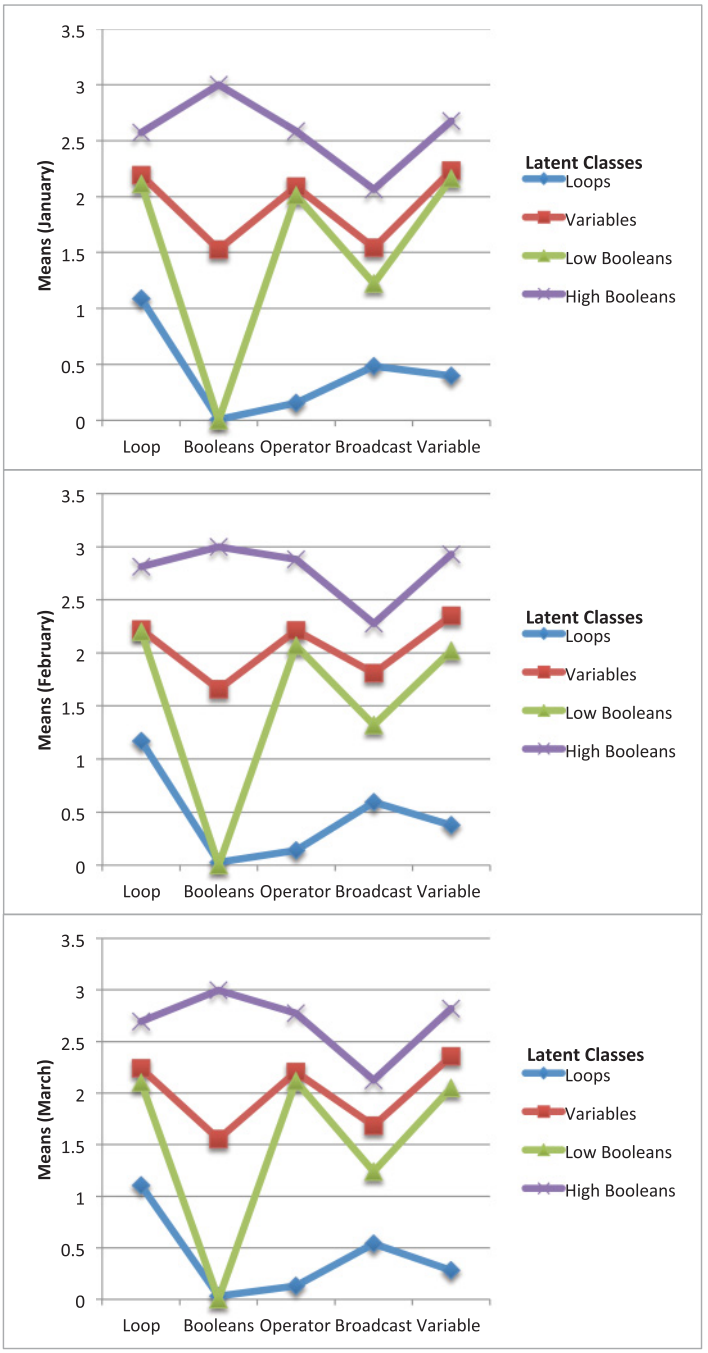

Fig. 3. LCA profile means for January, February, and March 2012, respectively.

Table 3. Description and Percentage of Users Classified in Each Programming Profile by Month

Programming Profile	Description	January (N = 1,719)	February (N = 902)	March (N = 679)
Class 1: Loops	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: Variables	More complex projects: Used all programming concepts except for Booleans	18.8%	16.2%	17.3%
Class 3: Low Booleans	Projects included all of the above plus Booleans	19.0%	14.5%	18.0%
Class 4: High Booleans	Projects used larger amounts of key concepts, especially Booleans	8.0%	11.2%	13.0%

members are consistent across time and situations, regardless of the number of users who created projects in a given month. We created titles for the groups based on some of the primary concepts used by an individual group (see Table 3).

By far the largest class of project creators (Class 1, Loops) used a minimal amount of loops (quartile 1, one to three loops) and almost no other programming concepts included in this analysis. These members seem to create relatively small and simple Scratch projects with few if any advanced kinds of commands. There is a relatively small likelihood that these members introduce operators, broadcasts, or variables, which may point to these commands being more difficult to use by many Scratchers. The second class of project creators (Class 2, Variables) introduced operators (i.e., conditionals) and variables into their Scratch programming projects and also utilized more loops (quartile 2, four to 11 loops) and events (using broadcast commands). We nicknamed this class with just one of the programming concepts (variables) purely for simplicity. The third and fourth classes both used Booleans (i.e., commands with "and," "or," and "not"). The main difference between the two is the number of Booleans and other concepts they use. Class 3 (Low Booleans) used roughly the same number of operators and variables as Class 2, but included Booleans in their projects. Class 4 (High Booleans) used more of everything, suggesting to us either that these projects are larger and more complex or possibly that these members simply create a larger number of projects with more of these programming concepts in a given month. Our naming of these groups as Loops, Variables, Low Booleans, and High Booleans may not be reflective of the actual sophistication of these members' programs. For instance, smaller projects with advanced commands (Class 3) might actually imply greater sophistication. The trends we relate later in this article further inform our understanding of who is in each class and how this changes (or not) over time.

5.2 Length of Membership Differences in Programming Profiles

Further analyses of the Scratchers' programming profiles revealed that membership, as indicated by account lifetime, matters somewhat (see Table 4). Not unexpectedly, our class of High Boolean Programmers (Class 4) are less likely to be newbies ($z = -2.0$) and more likely to be old-timers ($z = 2.1$). Confirming this trend, Variables programmers (Class 2) who utilized loops, events, variables, and conditionals in programming are less likely to be 1-year users ($z = -2.7$). Because these were

Table 4. Programming Profiles (Class) by Membership Count

	Account Lifetime			
Class	Newbies	Young	One-year	Oldies
Loops	384	283	179	138
Variables	117	111	34∼	51
Low Booleans	92	66	44	42
High Booleans	37∼	39	29	31 +

χ^2 (9) = 24.799, p = .003.
+ Standardized residual (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell.
∼ Standardized residual (z) is less than −2.0; this indicates a smaller proportion of individuals than expected for the cell.

Table 5a. Programming Profiles (Class) by Gender Count in January 2012

Class	Boy	Girls
Loops	575∼	409 +
Variables	186 +	58∼
Low Booleans	217	96
High Booleans	117 +	19∼

χ^2 (3) = 61.59, p < .001.
+ Standardized residual (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell.
∼z is less than −2.0; this indicates a smaller proportion of individuals than expected for the cell.

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 Loops and Low Boolean programmers, there are (proportionally) equal numbers of individuals across the four account lifetimes, and even within Loops (Class 1) and High Booleans (Class 4) programmers, there are Scratch members with a year or less of being registered on the site.

5.3 Gender Differences in Programming Profiles

In addition to length of membership in the online community, gender had a significant impact on which class Scratch programmers ended up in. We know that girls only represent one-third of all registered members on the Scratch site (as measured by self-reported gender) at the time of our study. The distribution in our overall sample reflected this distribution of self-reported gender on the Scratch site: 33% female and 67% male.[6] We tested whether gender was proportionately represented in each of the latent programming classes (see Table 5a). We found significant differences in regard to gender within three programming profiles, χ^2 (3) = 61.359, p < .001. Specifically, there were more girls than expected in Class 1 Loops (z = 3.7) and fewer girls than expected in Class 2 Variables (z = −2.9) and in Class 4 High Booleans (z = −4.1). Similarly, there were fewer boys than expected in the Loops class (z = −2.7) and more boys than expected in the Variables class (z = 2.1) and especially the High Booleans class (z = 3.0).

[6] As of April 2015, self-reported gender distribution on Scratch shifted to 38% female, 58% male, and 4% other/NA.

Table 5b. Programming Profiles (Class) by Gender Count in February 2012

Class	Boy	Girls
Loops	324~	191 +
Variables	102	43
Low Booleans	110 +	19~
High Booleans	89 +	12~~

Chi-square (3) = 42.60, p < .001.
+ Standardized residual (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell.
~z is less than −2.0; this indicates a smaller proportion of individuals than expected for the cell.

Table 5c. Programming Profiles (Class) by Gender Count in March 2012

Class	Boy	Girls
Loops	216	121 +
Variables	71	44
Low Booleans	99	18~
High Booleans	78 +	10~

Chi-square (3) = 38.55, p < .001.
+ Standardized residual (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell.
~z is less than −2.0; this indicates a smaller proportion of individuals than expected for the cell.

Across the following months, these differences continued, though to varying degrees (see Tables 5b and 5c). Class 1 (Loops) continued to show disproportionately high numbers of girls in both February and March, though the numbers were not quite as different in March. In Class 4 (High Booleans), there continued to be more boys and fewer girls in both February and March. Gender differences disappeared in Class 2 (Variables) in the later months but appeared in Class 3 (Low Booleans). Some of these changes may be due to shifting membership in each class (see next section) and to changing participation in Scratch—each month fewer and fewer individuals from the overall sample continued to share projects and participate in Scratch. Diminishing gender differences may suggest that members who continue to participate in sharing programs and participating in the Scratch site are more equal in terms of gender representation. Still, the disproportionately high number of girls in Class 1 (Loops) and the low number of girls in Class 4 (High Booleans) with the opposite patterns among boys show a problematic trend in terms of diversity in computational participation.

5.4 Relationship Between Programming Profiles Across Time

We used two approaches in latent class analyses and its posterior probabilities to examine the development of programming patterns across time. First, we examined how membership in qualitatively distinct programming groups changed across time in Tables 6a and 6b. Second, the correlations displayed in Table 7 examine the interrelationship among the use of particular programming patterns within and across time. Here we used pairwise correlations because of

Table 6a. Programming Profile Changes in Numbers and Percentage Between January and February (N = 591)

	Programming Profiles in February			
Programming Profiles in January	1	2	3	4
1. Loops	249 +	58	21∼∼	13∼
	73.0%	17.0%	6.2%	3.8%
2. Variables	26∼	21	23 +	18 +
	29.5%	23.9%	26.1%	20.5%
3. Low Booleans	51	22	18	10
	50.5%	21.8%	17.8%	9.9%
4. High Booleans	14∼	4∼	18 +	25 +
	23.0%	6.6%	29.5%	41.0%

χ^2 (9) = 155.852, p < .001.
+ Standardized residual (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell.
∼z is less than −2.0; this indicates a smaller proportion of individuals than expected for the cell.

Table 6b. Programming Profile Changes in Numbers and Percentage Between February and March (N = 396)

	Programming Profiles in March			
Programming Profiles in February	1	2	3	4
1. Loops	155 +	23	27∼	16∼
	70.1%	10.4%	12.2%	7.2%
2. Variables	30	21 +	9	7
	44.8%	31.3%	13.4%	10.4%
3. Low Booleans	16∼	9	19 +	15 +
	27.1%	15.3%	32.2%	25.4%
4. High Booleans	8∼	4	20 +	17 +
1. Loops	16.3%	8.2%	40.8%	34.7%

χ^2 (9) = 103.027, p < .001.
+ Standardized residual (z) is greater than 2.0; this indicates a greater proportion of individuals than expected for the cell.
∼z is less than −2.0; this indicates a smaller proportion of individuals than expected for the cell.

differences in sample size related to attrition issues and the fact that not all Scratch members created programs across all months. Both analyses show similar trends.

5.4.1 Class Movement. The first set of analyses (see Tables 6a and 6b) demonstrates a strong likelihood for members of Class 1 (Loops) to remain in Class 1 from January to February (73%) and similarly from February to March (70%). Similarly, there were proportionally fewer Class 1 members advancing to Classes 3 and 4 across time. This is the strongest trend among all the groups and the analyses of length of membership and gender suggest that this group is unique. It is strongly populated by newer Scratch members and disproportionately low in boys while high in girls. The fact that Scratch members who start in this group tend to stay in this group may suggest that it is a target for intervention in terms of outreach. From the least to most advanced, a relatively large

Table 7. Correlation Among Probabilities for Class Membership Across Months

	February				March			
January	1	2	3	4	1	2	3	4
1. Loops	.381***	−.022	−.260***	−.280***	.427***	−.066	−.249***	−.263***
2. Variables	−.238***	.061	.159***	.123**	−.204***	.035	.051	.180***
3. Low Booleans	−.072	.058	.055	−.017	−.159***	.089	.039	.085
4. High Booleans	−.240***	−.107**	.160***	.321***	−.224***	−.051	.287***	.084
	N (Jan & Feb)= 591				N (Jan & March) = 439			
March	1	2	3	4				
1. Loops	.407***	−.123*	−.226***	−.202***				
2. Variables	−.070	.220***	−.040	−.066				
3. Low Booleans	−.229***	.009	.140**	.154**				
4. High Booleans	−.279***	−.070	.227***	.207***				
	N (Feb & March) = 396							

*p < .05, **p < .01, ***p < .001.

percentage of Class 4 (High Booleans) programmers maintain the same membership status from January to February (41%) and from February to March (35%). Similarly, there were proportionally higher numbers of Class 4 members moving to Class 3 across time.

Beyond these trends, Classes 2, 3, and 4 show a great deal of movement between each other. From January to February, Class 2 shows a significant movement into Class 3 and 4; however, from February to March, no movement to those groups was found. Class 3 members were distributed proportionally evenly across the other classes in the same period. However, from February to March, Class 3 members were proportionally more likely to advance to Class 4 or remain in Class 3.

5.4.2 Correlation. The correlation results expand on some of these findings (see Table 7). Within time, the use of any particular programming patterns (Loops, Variables, Low Booleans, and High Booleans) was generally correlated with a decreased use of other programming patterns. For instance, the use of the Class 1 (Loops) programming pattern was correlated with the increased use of the same pattern and decreased uses of other patterns in both February and March.

Using Class 2 (Variables) programming patterns was connected to the use of more advanced patterns across time. For instance, using Class 2 programming in January was connected to increased use of Class 3 (Low Booleans) and Class 4 (High Booleans) programming in February and High Booleans programming in March. However, Low Booleans programming in January was not correlated to other programming patterns from January to February and March; there was some connection between February and March. Interestingly, High Booleans programming in January was significantly correlated with increased use of Low Booleans programming across time. Since the only difference between Class 4 (High Booleans) and Class 3 (Low Booleans) was the number of programming blocks rather than the variety of concepts used, this could simply mean that High Booleans users in January used the same concepts but contributed fewer or smaller projects in later months. From February to March, Class 2 (Variables) programming was only positively correlated with the continual use of the same program pattern. Low Booleans programming was connected to both the same program and use of High Booleans; this was the same pattern of High Booleans programming. This again shows how much members of Class 3 and Class 4 switch between programming patterns and may be due to volume of projects.

6 DISCUSSION

In this article, we examined computational participation in the wild, the type of self-organized and self-directed programming activities that have opened alternative paths for youth in the Scratch online community. What did we learn about Scratch programmers? We learned first that there are stable and cohesive classes of programmers in the Scratch online community that reflect a range of experience based on the use of particular programming concepts. In addition, we identified gender differences among several classes of programmers while ruling out main differences based on length of membership on the Scratch site. Finally, we studied Scratchers' membership in different classes of programming over time, identifying both stability and fluidity depending on the type of programming involved. In the absence of other studies that have examined the breadth of youth programming in an informal learning community, especially online communities, these findings present a first portrait of how self-directed learners in such online programming communities are distributed.

6.1 Nature and Equity of Computational Participation

One major contribution of our findings is to consider the range of novice programmers, rather than solely focusing on a small group of highly involved members who post dozens of projects (e.g., [35, 44, 48]). This allows us to consider patterns among Scratch members who are not as involved in the community. For instance, the one group of programmers who stayed surprisingly steady across all months was Class 1 (Loops), whose programs included only a few loops and no other more "advanced" concepts included in our analyses. Relatively few Scratch members in this group were likely to move to another class of programming that would have involved the use of programming concepts such as operators, synchronization/broadcasts, variables, and Booleans, generally considered to be more challenging for novice programmers. This Loops class also was disproportionately high in girls, a problematic finding given that the Scratch community is already predominantly male, though that has changed by a few percent in the past several years. The only other class that had consistent gender differences across time was Class 4 (High Booleans), the class that used the most of each programming concept in the study. Here the gender differences were reversed: there were more boys and fewer girls than expected in this class.

The stability of membership in Class 1 is also notable given the fluidity of membership in the other classes of programmers. Intriguingly, while the classes of programmers remained remarkably steady, even with drastically dropping membership with each progressive month, the membership in Classes 2, 3, and 4 (which might be considered home for the more advanced programmers) shifted each month, demonstrating some fluidity in the kinds of more advanced programming concepts used by participants in these classes. While Class 4 did show some signs of consistent membership (34%–40% of Scratch members were likely to remain in that class each month), in general there was movement between the Variables (Class 2), Low Booleans (Class 3), and High Booleans (Class 4).

One interpretation of this finding is that there are major differences between the participants in Class 1 (Loops) and the other classes, and that this is likely based on a limited knowledge of programming or possibly influenced by what kinds of projects members desire to create. Alternatively, this might also be an indicator of what kind of Scratch projects they do not want to create because it would require the use of synchronization or conditional logic. Given the disproportionate percentage of girls in Class 1, these findings suggest a potential target for intervention on the Scratch site.

It is helpful to put these findings into perspective and align them with our prior analyses of *participation* patterns on the Scratch website. Here we found almost no differences in regard to

gender and participation: each profile of Scratch participation was proportionately balanced in regard to gender [13]. In other words, while all Scratch members were equally involved in social networking activities, this was not the case in programming activities. We observed significant gender differences at the low and high poles of programming (Class 1 vs. Class 4). In particular, Class 4 (High Booleans) showed the only significant link to a particular participation profile (the High Networkers, those mostly likely to stay involved over time) in which there were no gender differences.

Why are there gender differences in programming profiles when there are no gender differences in participation profiles? 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 a better understanding of how social engagement might relate to programming engagement. As Ito et al. [25] 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 [33, 34], 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

One difficulty computer science educators face is evaluating learners' programming or computational thinking when projects are open ended, as in the Scratch community and many informal education sites [2]. Our work draws on the idea of identifying and measuring the use of various programming concepts by associating the use of individual scripts (or blocks) with a concept. While others have done the same (e.g., [8, 32, 48]), this approach is inevitably limited as it depends on frequency counts of blocks as a stand-in for learning. There is no way to see whether blocks are used correctly or functionally or to see nuances within programming concepts such as different approaches to events or variable use.

6.3 Future Directions

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 to address the challenges of analyzing not just the quantity but also the quality of beginners' programming. We already noted limitations in our logfile dataset 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

Moving ahead in learning analytics, we need to develop more sophisticated ways to study programming in open-ended learning environments, though these have yet to be applied to a community as broad and diverse as the Scratch site. For instance, Fields et al. [15] created an approach that looked for sets of blocks in context that represented functional use of concepts such as initialization, event-driven programming, or parallelism. Instead of just looking for blocks that are

used to create events in Scratch, they developed several measures of using events and developing parallelism where they could contextually identify whether this was done in a functional manner, how many were correct or incorrect, and whether these were used within one sprite (i.e., an object in Scratch) or across many sprites. From this they were able to study trajectories of learning that involved progressions within programming concepts [39] rather than measuring learning only by the number of concepts used. While this analysis was applied in the context of a computing camp organized around open-ended project challenges, this approach has yet to be tested in the context of even broader and less structured environments such as the Scratch online community.

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 [28]. 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 members 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 have been 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 long-standing differences in computing communities. 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. 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. Indeed, we should not assume that the Scratch users in our study were purely involved in Scratch "in the wild"—many of them likely had supports at home, in school, or from out-of-school programs. Mixed-methods research might be able to unveil some of the connections

between spaces in kids' trajectories of computational participation, comparing case study research with analytics derived from online or classroom sources as has been done with some virtual world research (see [11, 12, 28]). Rather than seeing them as isolated contexts, they can be harnessed for improved outreach and for better learning in mutually beneficial ways.

APPENDIX

Based on model indices (see Tables 8a to 8c), LCA results identify four classes of programming users. That is, large decreases and leveling off of model index values (aBIC, BIC, AIC) at and after the four-class model indicate that more complex models add little additional statistical depth. In addition, although LMR p-values and entropy values suggest that 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 1,719). For each participant, LCA generates probabilities for membership into each class, and generally one class has the highest probability of members. For

Table 8. Model Fit Indices for January to March

Table 8a. Model Fit Indices for January

January: N = 1,719							
# of Classes	Likelihood	Free Par	BIC	aBIC	LMR p-Value	Entropy	AIC
1	−13021.72	10	26,117.927	26,086.158	N/A	N/A	26,063.432
2	−11093.31	16	22,305.81	22,254.979	0.0000	0.948	22,218.618
3	−10152.07	22	20,468.028	20,398.137	0.0000	0.95	20,348.14
4	**−9699.059**	**28**	**19,606.705**	**19,517.752**	**0.0000**	**0.963**	**19,454.119**
5	−9494.66	34	19,242.602	19,134.588	0.0109	0.973	19,057.319
6	−8994.426	40	18,286.832	18,159.757	0.0112	0.999	18,068.853
7	−8935.328	46	18,213.332	18,067.196	0.0000	0.999	17,962.656

Table 8b. Model Fit Indices for February

February N = 902							
# of Classes	Likelihood	Free Par	BIC	aBIC	LMR p-Value	Entropy	AIC
1	−7057.718	10	14,183.482	14,151.724	NA	NA	14,135.436
2	−5804.549	16	11,717.971	11,667.157	0.0000	0.967	11,641.097
3	−5244.767	22	10,639.236	10,569.368	0.0001	0.974	10,533.535
4	**−5032.713**	**28**	**10,255.955**	**10,167.032**	**0.0034**	**0.976**	**10,121.426**
5	−4885.536	34	10,002.428	9,894.449	0.0002	0.983	9,839.071
6	−4707.744	40	9,687.673	9,560.64	0.4797	0.952	9,495.489
7	−4485.267	46	9,283.546	9,137.457	0.0072	0.999	9,062.533

Table 8c. Model Fit Indices for March

March N = 679							
# of Classes	Likelihood	Free Par	BIC	aBIC	LMR p-value	Entropy	AIC
1	−5355.246	10	10,775.697	10,743.946	NA	NA	10,730.491
2	−4375.486	16	8,855.302	8,804.5	0.0000	0.974	8,782.972
3	−4033.562	22	8,210.577	8,140.724	0.0011	0.966	8,111.123
4	**−3902.107**	**28**	**7,986.791**	**7,897.888**	**0.0028**	**0.970**	**7,860.214**
5	−3789.717	34	7,801.134	7,693.181	0.0001	0.979	7,647.433
6	−3674.031	40	7,608.886	7,481.882	0.0080	0.997	7,428.061
7	−3554.136	46	7,408.221	7,262.166	0.0759	0.997	7,200.272

Table 9a. 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**

instance, results show that the January users classified into their highest-probability latent class had a 95% to 100% probability of being in that class and a 0% to 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. Tables 8a to 8c display the fit indices for January, February, and March.

ACKNOWLEDGMENTS

This material is based on 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, Xavier Velasquez, and Whitney King for reading earlier versions of this article. Particular thanks to Anant Seethalakshmi and Andres Monroy-Hernández for help with gathering data and to the Scratch Team for providing repeated feedback on analysis.

REFERENCES

[1] C. Aragon, S. Poon, A. Monroy-Hernández, and D. Aragon. 2009. A tale of two online communities: Fostering collaboration and creativity in scientists and children. In *Proceedings of the Creativity and Cognition Conference*. New York: ACM Press, 9–18.

[2] M. Berland, R. S. Baker, and P. Blikstein. 2014. Educational data mining and learning analytics: Applications to constructionist research. *Technology, Knowledge and Learning* 19, 1–2, 205–220.

[3] M. Berland, T. Martin, T. Benton, C. P. Smith, and D. Davis. 2013. Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences* 22, 4, 564–99.

[4] P. Blikstein, M. Worsley, C. Piech, M. Sahami, S. Cooper, and D. Koller. 2014. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences* 23, 4, 561–599.
[5] K. Brennan. 2013. Best of Both Worlds: Issues of Structure and Agency in Computational Creation, in and out of School. Unpublished Dissertation. Cambridge, MA: Massachusetts Institute of Technology.
[6] K. Brennan and M. Resnick. 2012. New Frameworks for Studying and Assessing the Development of Computational Thinking. Paper presented at annual American Educational Research Association meeting. Vancouver, BC, Canada.
[7] A. S. Bruckman. 1997. MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids. Unpublished Dissertation. Cambridge, MA: Massachusetts Institute of Technology.
[8] S. Dasgupta, W. Hale, A. Monroy-Hernández, and B. M. 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)*. New York: ACM Press, 1438–1449.
[9] J. Denner, L. Werner, S. Campe, and E. Ortiz. 2014. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education* 46, 3, 277–296
[10] D. A. Fields, M. T. Giang, and Y. B. Kafai. 2014. Programming in the wild: Patterns of computational participation in the Scratch online social networking forum. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE'14)*. ACM, New York, 2–11. http://doi.acm.org/10.1145/2670757.2670768
[11] D. A. Fields and Y. B. Kafai. 2009. A connective ethnography of peer knowledge sharing and diffusion in a tween virtual world. *International Journal of Computer Supported Collaborative Learning* 4, 1, 47–68.
[12] D. A. Fields and Y. B. Kafai. 2010. Knowing and throwing mudballs, hearts, piece, and flowers: A connective ethnography of gaming practices. *Games and Culture* 5, 1, 43–63.
[13] D. A. Fields, Y. B. Kafai, and M. T. Giang. 2016. Participation by choice: A transitional analysis of patterns in social networking and coding contributions in the online Scratch community. In *Mass Collaboration and Education*. U. Cress, H. Jeong, and J. Moskaliuk (Eds.). New York: Springer, 209–240.
[14] D. A. Fields, K. Pantic, and Y. B. Kafai. 2015. "I have a tutorial for this": The language of online peer support in the Scratch programming community. In *Proceedings of Interaction Design and Children (IDC'15)*. ACM, New York, 229–238.
[15] D. A. Fields, L. Quirke, J. Amely, and J. Maughan. 2016. Combining big data and thick data analyses for understanding youth learning trajectories in a Summer Coding Camp. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 150–155.
[16] D. A. Fields, V. Vasudevan, and Y. B. Kafai. 2015. The programmers' collective: Fostering participatory culture by making music videos in a high school Scratch coding workshop. *Interactive Learning Environments* 23, 5, 1–21.
[17] N. F. Velasquez, D. A. Fields, D. Olsen, H. Taylor Martin, A. Strommer, M. C. Sheperd, and Y. B. Kafai. 2013. Novice programmers talking about projects: What automated text analysis reveals about online Scratch users' comments. In *Proceedings of the 2014 47th Hawaii International Conference on System Sciences (HICSS'14)*. IEEE Computer Society, Washington, DC, 1635–1644.
[18] J. P. Gee. 2004. *Situated Language and Learning: A Critique of Traditional Schooling*. New York and London: Routledge.
[19] S. M. Grimes and D. A. Fields. 2012. *Kids Online: A New Research Agenda for Understanding Social Networking Forums*. The Joan Ganz Cooney Center at Sesame Workshop, New York. Available online at http://www.joanganzcooneycenter.org/reports-38.html.
[20] S. M. Grimes and D. A. Fields. 2015. Children's media making, but not sharing: The potential and limitations of child-specific DIY media websites for a more inclusive media landscape. *Media International Australia* 154, 112–122.
[21] S. Grover and R. Pea. 2013. Computational thinking in K–12 a review of the state of the field. *Educational Researcher* 42, 1, 38–43.
[22] M. Guzdial. 2015. *Learner-Centered Design of Computing Education: Research on Computing for Everyone*. San Rafeal, CA: Morgan & Claypool.
[23] J. Hagenaars and A. McCutcheon (Eds). 2002. *Applied Latent Class Analysis*. Cambridge, UK: Cambridge University Press.
[24] B. M. Hill and A. Monroy-Hernández. 2017. A longitudinal dataset of five years of public activity in the Scratch online community. *Scientific Data* 4:170002.
[25] 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. 2010. *Hanging Out, Messing Around, and Geeking Out: Living and Learning with New Media*. Cambridge, MA: MIT Press.
[26] Y. B. Kafai. 1995 *Minds in Play: Computer Game Design as a Context for Children's Learning*. New York: Routledge.
[27] Y. B. Kafai and W. Quinn Burke. 2014. *Connected Code: Why Children Need to Learn Programming*. Cambridge, MA: MIT Press.
[28] Y. B. Kafai and D. A. Fields. 2013. *Connected Play: Tweens in a Virtual World*. Cambridge, MA: MIT Press.

[29] Y. B. Kafai, D. A. Fields, R. Roque, W. Quinn Burke, and A. Monroy-Hernández. 2012. Collaborative agency in youth online and offline creative production in Scratch. *Research and Practice in Technology Enhanced Learning* 7, 2, 63–87.
[30] Y. F. Kafai, K. A. Peppler, and R. N. Chapman. 2009. *The Computer Clubhouse: Constructionism and Creativity in Youth Communities*. New York: Teachers College Press.
[31] C. Kelleher and R. Pausch. 2007. Using storytelling to motivate programming. *Communications of the ACM* 50, 7, 58–64.
[32] J. H. Maloney, K. Peppler, Y. B. Kafai, M. Resnick, and N. Rusk. 2008. Programming by choice: Urban youth learning programming with scratch. *ACM SIGCSE Bulletin* 40, 1, 367–371.
[33] J. Margolis, R. Estrella, J. Goode, J. Holme, and K. Nao. 2008. *Stuck in the Shallow End: Education, Race, and Computing*. Cambridge, MA: MIT Press.
[34] J. Margolis and A. Fisher. 2002. *Unlocking the Clubhouse*. Cambridge, MA: MIT Press.
[35] J. N. Matias, S. Dasgupta, and B. M. Hill. 2016. Skill progression in scratch revisited. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. ACM, New York, 1486–1490.
[36] A. Monroy-Hernández. 2012. Designing for Remixing: Supporting an Online Community of Amateur Creators. Unpublished Dissertation. Cambridge, MA: MIT.
[37] B. Muthen. 2002. Statistical and substantive checking in growth mixture modeling. Retrieved January 2007 from http://www.gseis.ucla.edu/faculty/muthen/full_paper_list.h.
[38] B. Muthen and L. Muthen. 2001. Integrating person-centered and variable-centered analyses: Growth mixture modeling with latent trajectory classes. *Alcohol Clinical Experimental Research* 24, 6, 882–891.
[39] K. Pantic, D. A. Fields, and D. A. Lisa Quirke. 2016. Studying situated learning in a constructionist programming camp: A multimethod microgenetic analysis of one girl's learning pathway. In *Proceedings of the 15th International Conference on Interaction Design and Children*. ACM, 428–439.
[40] L. Porter, M. Guzdial, C. McDowell, and B. Simon. 2013. Success in introductory programming: What works? *Communications of ACM* 56, 8, 34–36.
[41] A. Repenning, D. C. Webb, K. H. Koh, H. Nickerson, S. B. Miller, C. Brand, I. Her Many Horses, A. Basawapatna, F. Gluck, R. Grover, K. Gutierrez, and N. Repenning. 2015. Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computer Education* 15, 2, 265–269.
[42] M. Resnick, J. Maloney, A. Monroy-Hernańdez, N. Rusk, E. Eastmond, K. Brennan, A. D. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. B. Kafai. 2009. Scratch: Programming for everyone. *Communications of the ACM* 52, 11, 60–67.
[43] G. T. Richard and Y. B. Kafai. 2016. Blind spots in youth DIY programming: Examining diversity in creators, content and comments within the Scratch online community. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1473–1485.
[44] C. Scaffidi and C. Chambers. 2012. Skill progression demonstrated by users in the Scratch animation environment. *International Journal of Human-Computer Interaction* 28, 6, 383–398.
[45] E. Soloway and J. Spohrer. 1990. *Empirical Studies of Novice Programmers*. Norwood, NJ: Ablex Publishing.
[46] L. Werner, C. McDowell, and J. Denner. 2013. A first step in learning analytics: Pre-processing low-level Alice logging data of middle school students. *Journal of Educational Data Mining* 5, 2, 11–37.
[47] White House. January 30, 2016. FACT SHEET: President Obama announces computer science for all initiative. Washington, DC: Office of the Press Secretary. Available at: https://www.whitehouse.gov/the-press-office/2016/01/30/fact-sheet-president-obama-announces-computer-science-all-initiative-0
[48] S. Yang, C. Domeniconi, M. Revelle, M. Sweeney, B. U. Gelman, C. Beckley, and A. Johri. 2015. Uncovering trajectories of informal learning in large online communities of creators. In *Proceedings of the 2nd (2015) ACM Conference on Learning @ Scale (L@S'15)*. ACM, New York, 131–140.

Received October 2016; revised March 2017; accepted May 2017