Learning to Code in Localized Programming Languages

Sayamindu Dasgupta[*†]
[*]MIT Media Lab
Cambridge, MA 02142
sayamindu@media.mit.edu

Benjamin Mako Hill[†]
[†]University of Washington
Seattle, WA, 98195
{sdg1, makohill}@uw.edu

ABSTRACT

Education research suggests that learning in one's local language can have a positive impact on learning outcomes. We offer a quantitative test of the association between local language use and the rate at which youth learn to program. Using longitudinal data drawn from five countries and over 15,000 users of Scratch, a large informal learning community, we find that novice users who code with their programming language keywords and environment localized into their home countries' primary language demonstrate new programming concepts at a faster rate than users from the same countries whose interface is in English. We conclude with a discussion of the implications of our findings for designers of online learning systems.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g. HCI): User Interfaces; K.3.2 Computer and Information Science Education: Computer Science Education

Author Keywords

learning; programming language education; localization; learning in local languages; linguistic accessibility; Scratch

INTRODUCTION

A large body of education research and theory suggests that learning in one's local language at the primary and secondary levels supports positive learning outcomes [20, 4, 13]. As early as 1953, UNESCO's publication on "the use of vernacular languages in education" [20] argued for instruction in students' mother tongues both as early, and as late, as possible:

> On educational grounds, we recommend that the use of the mother tongue be extended to as late a stage in education as possible. In particular, pupils should begin their schooling through the medium of the mother tongue because they understand it best [. . .]

A 2008 study that covered 26 countries and 153 linguistic groups found that attendance in educational institutions was positively related with the availability of mother-tongue instruction [18]. Another study in Guatemala showed that schools that offered bilingual education in students' mother tongues have higher attendance and promotion rates and lower repetition and dropout rates [12].

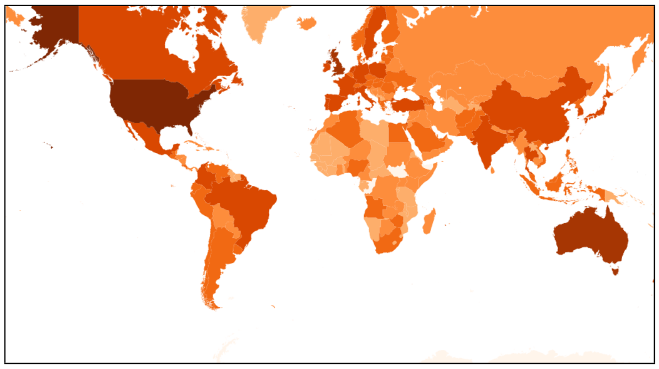
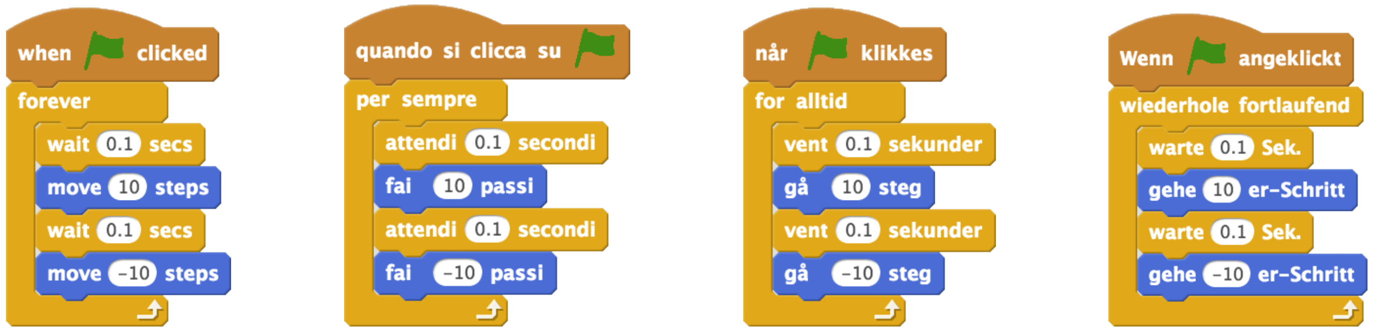


Figure 1: Choropleth map showing the distribution of Scratch users worldwide as indicated by self-reported country information. Countries are shaded from lightest to darkest in terms of the number of Scratch users in the country along a log-linear scale [10].

As the audiences for large-scale digital learning tools and environments become more global, the potential benefit of supporting local languages in these systems has grown. Despite prior work in HCI highlighting the need for localized[1] user interfaces [15, 9], localization can be complex and expensive [7]. As a result, many online learning systems are offered only in English. When interfaces are localized, it is often only into a small number of languages spoken by large populations. For speakers of other languages, especially for those belonging to small or marginalized linguistic groups, the only option is to use English.

The use of English is particularly pronounced in programming education. In nearly every widely used programming language, the names of symbols and keywords (e.g., for, while, if, else) are borrowed from English. In this paper, we examine the relationship between the rate at which young people learn to code and their use of localized programming languages and user interfaces. Using longitudinal data on the learning trajectories of 15,015 users of the Scratch online community from five countries, we find that, controlling for differing

[1]In this paper we use the term "localization" to mean translated user interfaces. However, localization often consists of more than translation—for example, customized UI icons, customized representations of numbers, locale appropriate units, support for local calendars, and so on.



```
when [flag] clicked
forever
    wait 0.1 secs
    move 10 steps
    wait 0.1 secs
    move -10 steps
```

(a) English

```
quando si clicca su [flag]
per sempre
    attendi 0.1 secondi
    fai 10 passi
    attendi 0.1 secondi
    fai -10 passi
```

(b) Italian

```
når [flag] klikkes
for alltid
    vent 0.1 sekunder
    gå 10 steg
    vent 0.1 sekunder
    gå -10 steg
```

(c) Norwegian Bokmål

```
Wenn [flag] angeklickt
wiederhole fortlaufend
    warte 0.1 Sek.
    gehe 10 er-Schritt
    warte 0.1 Sek.
    gehe -10 er-Schritt
```

(d) German

Figure 2: Functionally identical Scratch code represented in four different languages.

levels of activity and socialization, learners who code with localized programming language keywords and environments demonstrate new programming concepts at a faster rate relative to users from the same countries who use otherwise identical interfaces in English. We conclude with a discussion of our findings, the relatively small size of our estimated effect, and implications of our study for designers of online learning systems.

Our empirical setting is Scratch [14], a programming language and online community designed for children aged 8-16. In Scratch, programs are constructed by dragging and dropping visual blocks to define the behavior of on-screen graphical objects called sprites (Figure 2). The Scratch language is supported by a large online community launched in 2007 [11], where creators can share their projects, comment on each others' work, and remix projects created by their peers. Although participation in Scratch is generally open-ended and self-directed, Scratch is used in formal educational settings as well [2].

When Scratch users register they are asked to list their home country. Scratch's 16 million users come from all over the world (Figure 1). The Scratch programming editor, website, and programming blocks are translated into a number of languages by volunteer translators through a web-based translation portal. When a user visits the Scratch website, the interface language is automatically selected based on the user's browser configuration. In previous studies, this mechanism has been used to identify English as a Second Language (ESL) users in MOOCS [19]. Scratch users can also select their language manually from a menu in the website or code-editor interface. The language choice persists across browser sessions through a HTTP cookie-based mechanism. In exploratory analysis, we found that a majority of the users (approx. 88%) use a single language throughout their interactions with the site.

In Scratch, a single piece of code can be represented in different languages depending on the language selected by the user. Figure 2 shows the same Scratch code, from the same project, represented in four different languages. For example, when a user of the localized Japanese interface creates a project and shares it, they do so entirely in Japanese. If a user of the Italian localized interface were to view or edit the Japanese user's project, the code would appear in Italian.

Scratch reflects a unique source of observational data to answer questions about the effect of localization on learning for several reasons: it is among the largest websites where young people learn to program, it is used by large numbers of users around the world, and it has been localized into dozens of languages by a large team of volunteer translators. Given education research that suggests that local language use will lead to better learning outcomes, our hypothesis is that *Scratch users will learn programming concepts more rapidly when their programming language keywords and environments are localized into their own languages*.

DATA AND MEASURES

Blocks in Scratch are the equivalent of tokens in text-based programming languages and can be used to make a character on the screen (sprite) move, to change a variable, to repeat a set of instructions, and so on. For example, in Figure 2, blocks for making a sprite move back and forth are enclosed in an outer "forever" loop block that is connected to an event-handler block ("when green flag clicked"). In previous research on Scratch and other block-based programming languages such as App Inventor, learning has been modeled as growth in the cumulative repertoire of blocks, similar to a measure of demonstrated vocabulary, which may grow more or less quickly over time [22, 5, 21]. We follow a similar approach and construct a longitudinal measure that represents a cumulative block repertoire over time for each user in our dataset. The cumulative block repertoire is updated for every *de novo* (i.e., non-remix) project shared and incremented by the number of types of blocks in each project that the user had never used previously. For example, if a given project uses two instances of the `if` block, and if the user has not used the `if` block before in previous projects, we increase the cumulative block repertoire by one. We use this measure ($\textit{Cumulative Block Repertoire}_{up}$) as our measure of learning and as the dependent variable in our analysis. The subscript "*up*" reflects the fact that the variable is at the level of the project, clustered within user.

A central independent variable in our analysis is a measure of whether users are interacting with Scratch using a localized interface. To construct this variable, we seek to exploit varia-

Country	Language	% average translation	# users	% users w/ local language	# projects
Portugal	Portuguese	98.9	2,630	90.2	13,178
Italy	Italian	98.5	3,820	87.3	19,875
Brazil	Brazilian Portuguese	96.5	4,353	88.0	26,170
Germany	German	95.8	2,666	75.2	14,339
Norway	Norwegian Bokmål	95.5	1,546	84.6	6,372

Table 1: Countries included in the dataset used in this paper, along with the corresponding primary language, average translation level, number of users, proportion of users who use Scratch in the local language, and total number of projects (with usable language data) per country.

tion in the agreement between users' local language and their interface language. For example, we seek to compare Italian-speaking youths using Italian interfaces to Italian-speaking youths using non-localized English interfaces. Constructing this variable using observation data means we must separately infer each user's primary language and interface language.

There are several reasons why a user might use an interface in a language other than their primary language. One set of reasons is related to the fact that the Scratch website attempts to infer users' language preference on their first visit using metadata sent by the users' web browser. If a user's computer is not configured to send language preference data when a user first visits Scratch, or if any number of technical issues prevent this from happening correctly, users will end up using Scratch in English. When this happens, users may not change the language because they are not aware of the existence of the menu for switching their interface language manually. In another scenario, a learner who has a working knowledge of English may choose to use the system in English because of external factors (e.g., the instructional material being followed may be in English).

Identifying a users' interface language in Scratch is relatively straightforward. When a Scratch user creates a project, the language choice of the user at that moment is recorded. We use this information to determine each user's interface language. Because a large majority of users (88%) never change their language, we set each user's language to what they have used for more than 50% of their published *de novo* projects.

A Scratch user's primary language, independent of their interface, is not possible to observe directly. As a proxy, we first identify the countries that users' self-report as their home at the time that they create their accounts. We then associate languages with countries using *Ethnologue: Languages of the World* [8] to identify the most widely-spoken languages in each country. We treat these national languages as users' primary language. At this point, we excluded all users from countries where English is the mostly widely spoken language as well as users from countries without a single language spoken by a majority of the population.

Since translation in Scratch is volunteer driven, there are only a few languages into which the Scratch website, the Scratch code editor, and the Scratch programming blocks are fully translated. Additionally, the Scratch website and the editor are being continuously developed and updated with bug fixes and new features. Frequently, these updates cause translations to become out of date, which causes coverage to vary over time. Because we are interested in the effect of translated interfaces, we attempt to identify a set of languages into which the Scratch interface (website, code-editor, and programming blocks) was fully translated. Toward this end, we calculate the translation coverage in all languages twice, at the beginning of our data-collection period and at the end. Although no languages were 100% translated at both points, five languages had at least 95% average translation coverage: Portuguese, Italian, Brazilian Portuguese, German and Norwegian Bokmål. As a result, we restrict our analysis to users who reported their home countries as Portugal, Italy, Brazil, Germany, and Norway, respectively.

Using these measures of users' inferred interface and primary language, we construct a dummy variable ($Localized?_u$ with a subscript u because it does not vary across users) set to 1 if a user's inferred interface language matches their home country's most widely used language (i.e., Portuguese, Italian, Brazilian Portuguese, German, and Norwegian Bokmål, respectively). Table 1 shows the number of users from each country in the resulting dataset who use Scratch in their local language. As an additional robustness check of our study, we did a similar analysis with languages above the 90% translation level threshold with two additional countries (France and Slovenia). The overall results remained similar in magnitude and sign. Next, we construct a continuous variable that represents the cumulative number of *de novo* projects shared by the user prior to the observation ($De\ Novo\ Projects_{up}$). The interaction between these two variables constitutes our key question predictor.

We also construct a series of controls used by Dasgupta *et al.* [5]. We construct a control for the cumulative number of blocks used in users' *de novo* projects ($Total\ Blocks_{up}$). To control for social activity, we construct a control for the number of comments that projects' creators have left on Scratch at the time that they shared the project ($Comments_{up}$). To capture the effect of remixing, we construct a control for users' numbers of remixes ($Remixed\ Projects_{up}$). Dasgupta *et al.*, whose dataset was drawn from an earlier version of Scratch, used a count of downloads as a measure of users' exposure to other users' source code. In the newer version of Scratch used in our analysis, downloading has been replaced with 'See Inside" functionality that allows users to view source code directly on the website. Hence, we add a control for the number of times users have seen inside another user's projects (*See Insides$_{up}$*). Finally, we construct controls for users' tenures on the site in days (*Experience$_{up}$*) and self-reported age measured in years (*User Age$_{up}$*).

For our analysis, we constructed a dataset that contains all projects shared by users from the five focal countries who signed up between May 15, 2015 and May 15, 2016. We chose May 15, 2015 as the data collection start date as the Scratch software started to log language information just prior to that date. As we rely on longitudinal measures for our study, we consider only those users who have shared more than one *de novo* project during our period of analysis. Out of a total of 15,339 users from this period who shared more than one *de novo* project, we discard 324 who were found to have used Scratch in a language that is neither the local language of that country nor English. Our final dataset includes a total of 90,859 projects shared between May 15, 2015 to July 8, 2016 by the remaining 15,015 users.

Variable	Range	μ	M	σ
Cumulative Block Repertoire	[0, 139]	27	25	18
De Novo Projects	[1, 744]	6	3	12
Remixed Projects	[0, 570]	1	0	7
Comments	[0, 2771]	4	0	41
Total Blocks	[0, 72461]	487	138	1909
Experience	[0, 415]	62	39	71
User Age	[4, 89]	19	15	12

Table 2: Summary statistics for users in our dataset at the point of the final project included in the dataset. Columns report variables' range, mean (μ), median (M), and standard deviation (σ).

Among these projects, 10,768 are missing language information. Another 157 projects are missing all data in the Scratch database. Examination of randomly chosen projects where language information is missing suggests that these were projects uploaded from the standalone Scratch offline client, which though localized, does not include language information with project uploads. For the 157 projects without any data at all, it is hard to pinpoint a reason that data is missing. One possibility is that the Scratch website's project upload or auto-save mechanism failed for these projects. As an additional data cleanup measure, we mark 2,389 projects as having missing age information, where the user age was found to be self-reported as less than 4 years, or more than 90 years of age. As with any online-activity dataset all our variables of interest are highly skewed, and we provide summary statistics for each user in our dataset (post cleanup) in Table 2 at the point when users shared their final projects.

ANALYTIC PLAN

Our analytic plan closely follows the approach used by Dasgupta *et al.* Our dataset is structured so that our unit of analysis is the *de novo* Scratch project. Of course, measures of cumulative block repertoire within a particular user are not independent of each other, and this introduces an important threat of serial correlation of standard errors. The most common technique for addressing this threat is the use of multi-level models [17]. Dasgupta *et al.* [5] addressed this with user-level fixed effects, which control for all observed and unobserved qualities that have a consistent effect across projects shared by a user. This strategy is not possible in this study because a key question predictor (*Localized?$_u$*) does not vary across users' projects. Instead, we include a random intercept term. Since our dependent variable is a count, we model growth in block repertoire using mixed-effects Poisson regression models using the lme4 package in R [1]. As we are concerned about over-dispersion in the dependent variable, we fit a negative binomial regression model as well, and the results are similar. Since the distributions of the continuous independent variables are skewed, we log-transform all of them except for *User Age$_{up}$*.

In our first model (M1), we consider only three variables. First, we include our dummy variable, which indicates whether the user in question is creating projects with a localized interface (*Localized?$_u$*). Second, we include our continuous variable that represents the cumulative number of shared *de novo* projects (*De Novo Projects$_{up}$*). Finally, because our hypothesis is that the growth of repertoires for users using localized interfaces will be faster, we include an interaction term between *Localized?$_u$* and *De Novo Projects$_{up}$*, which allows us to estimate the difference in slope associated with a localized interface. The formal version of M1 is presented below:

$$\begin{aligned}\text{Cumulative Block Repertoire}_{up} &= \beta_0 + \\ \beta_1 \ln \text{Localized?}_u + \beta_2 \ln \text{De Novo Projects}_{up} &+ \\ \beta_3 \text{Localized?}_u \times \ln \text{De Novo Projects}_{up} &+ [u_{0i} + \varepsilon_{up}]\end{aligned}$$

In our second model (M2), we add the series of controls described above. Formally, M2 can be represented as:

$$\begin{aligned}\text{Cumulative Block Repertoire}_{up} &= \beta_0 + \\ \beta_1 \text{Localized?}_u + \beta_2 \ln \text{De Novo Projects}_{up} &+ \\ \beta_3 \text{Localized?}_u \times \ln \text{De Novo Projects}_{up} & \\ + \beta_4 \ln \text{Remixed Projects}_{up} + \beta_5 \ln \text{See Insides}_{up} &+ \\ \beta_6 \ln \text{Comments}_{up} + \beta_7 \ln \text{Total Blocks}_{up} &+ \\ \beta_8 \ln \text{Experience}_{up} + \beta_9 \text{User Age}_{up} &+ [u_{0i} + \varepsilon_{up}]\end{aligned}$$

RESULTS

Shown in Table 3, the results of parameter estimates for both models suggest a positive effect of localization on the rate at which block repertoire increases. Because the addition of controls in M2 does not alter our results substantively, we discuss only the results from M2 below. The main effect of *Localized?* is negative ($\beta = -0.056$, $SE = 0.015$, $p < 0.01$). This suggests that, everything else equal, the first projects by users of the localized interface have a lower diversity of blocks. Given that a user's cumulative repertoire cannot decrease, it is unsurprising that we estimate that the main effect of *De Novo Projects* is positive ($\beta = 0.012$, $SE = 0.005$, $p < 0.001$).

	Cumulative Block Repertoire	
	M1	M2
Localized?$_u$	-0.109^*	-0.056^*
	(0.020)	(0.015)
ln De Novo Projects$_{up}$	0.517^*	0.012^*
	(0.003)	(0.005)
Localized?$_u$ $\times$ ln De Novo Projects$_{up}$	0.091^*	0.027^*
	(0.004)	(0.004)
ln Remixed Projects$_{up}$		-0.033^*
		(0.002)
ln See Insides$_{up}$		0.020^*
		(0.002)
ln Comments$_{up}$		-0.048^*
		(0.002)
ln Total Blocks$_{up}$		0.371^*
		(0.002)
ln Experience$_{up}$		0.052^*
		(0.001)
User Age$_{up}$		-0.001^*
		(0.0004)
Constant	2.119^*	1.045^*
	(0.019)	(0.016)
Observations	79,934	77,545
Log Likelihood	-284,220.000	-246,430.700
Note:		*p<0.01

Table 3: Results of fitting mixed-effects Poisson regression models M1 and M2.

Our key question predictor is the interaction term between *De Novo Projects* and *Localized?*, which captures the difference in the rate of repertoire growth between users of localized and English interfaces. Our estimates are small but positive ($\beta = 0.027$, $SE = 0.004$, $p < 0.001$). Although we estimate that users of non-localized interfaces begin with higher repertoires, our model predicts that users who have shared 2 log units more projects will have the same repertoire as otherwise identical users of non-localized interfaces and that this gap will widen as users share more.

Because our model is a non-linear model, because most of our our independent variables are log-transformed, and because our key predictor is an interaction, interpreting the coefficients in our model directly can be challenging. We find it useful to interpret the results by describing model-predicted values for "prototypical" users. For example, M2 predicts that a user who has shared 16 projects (95th percentile in our dataset) with median values for all of our control variables would have a repertoire that would be 37.1 if they had used English but 37.9 if they had used a localized interface instead. Though the difference of less than 1 block is small, the negative main effect of *Localized?$_u$* masks some of the differences in the rate of change.

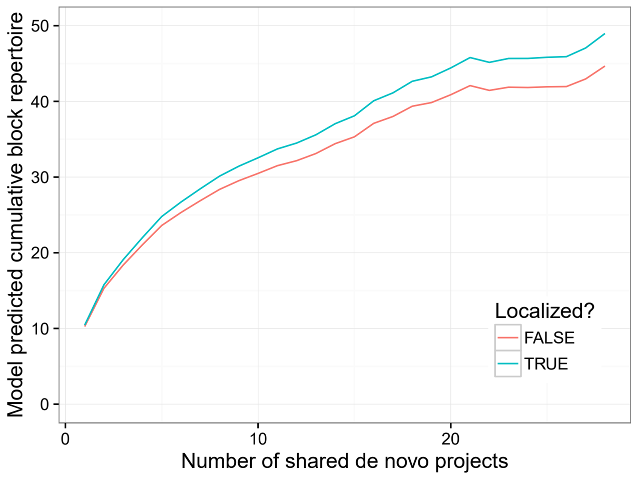


Figure 3: Model-derived predicted values (from model M2) for *Cumulative Block Repertoire* for two prototypical users across a range of values of *de novo* projects. The main effect of localized language use is set to zero. All other predictor variables are set to median values for the corresponding value of *de novo* project share count.

To further aid in interpretation, we visualize a range of these model-predicted values drawn from M2 in Figure 3 for two such prototypical users: one who uses a localized interface and the other who does not. The x-axis represents the number of *de novo* projects shared (up to the 98th percentile in our dataset), and the y-axis represents the users' model-predicted cumulative block repertoire. Since all other predictor variables tend to increase over time, we set each of of our controls to the median value across all projects with the same value of *De Novo Projects* (e.g., the median for users' 1st, 2nd, 3rd, ... etc. projects). To make it easier to interpret the effect of the interaction, we also set the main effect of *Localized?* to 0 so that both prototypical users start with identical repertoires. A user who has shared 16 projects (95th percentile in our dataset) with median values for all of our control variables would have a repertoire that would be 37.1 if they had used English but 40.1 if they had used a localized interface instead.

All of the parameter estimates for our controls are well estimated. Drawn from a very different version of Scratch and from a non-US sample, several of our estimates for our controls are different in sign from those found in the earlier study by Dasgupta *et al.* [5]. Most important for the previous work, Dasgupta *et al.* found a positive relationship between remixing activity and learning, and we estimate a negative association. Although this might be explained by revisions to Scratch that changed the way that code is shared and reused, our model points to an important area for further study. In terms of our papers' findings, the addition of the controls in M2 attenuate the size of our effect but does not alter the sign or substantive takeaway.

LIMITATIONS

Of course, the validity of our findings and results could be affected by a number of potential threats to validity. First, cumulative block repertoire can increase at a faster rate for reasons unrelated to, but correlated with, users' use of a localized interface. In other words, although our results can be understood as evidence in support of the hypothesis that local language use can cause users to learn about new programming blocks more quickly, our results describe correlation, not causation. We have attempted to address this threat by including a number of variables in our model as controls for productivity, social activity, and age, but there may be other important variables that we have omitted. For example, wealthier individuals are often more likely to be more fluent with English [6]. If relatively wealthier users within a country are more likely to use a non-localized interface, they might also learn faster or slower than their less wealthy peers for reasons other than the language of the Scratch interface.

Second, it is unclear what effect the English-dominated nature of the Scratch online community has on learning. In our analysis, we choose languages where translation coverage is very high, meaning that the website is localized almost completely. However, the content on the Scratch website includes prompts within projects, comments, and project descriptions. Based on our unscientific observations, most of this material—with the exception of language specific forums—is in English. If social learning within Scratch is supported by English fluency, we might expect English users to be at a relative advantage compared to users relying on localization. Ultimately, however, the implication of this is unknown.

Third, our analysis relies on a series of assumptions that we know are not always true. Central to our construction of our variable *Localized?* is the assumption that everyone in each country in our dataset speaks the same language (e.g., that in Germany, everyone speaks German). This is clearly not always true. In future work, it may be possible to use language-detection algorithms [16] to infer users' primary languages from their submitted content (e.g., project titles, comments, variable names in projects). However, the presence of English-speaking users in our dataset using the non-localized interfaces seems most likely to lead to under-estimates of the effect of localization on learning because these English-speaking users in non-English-speaking countries would be using their preferred language and would not be at a relative disadvantage.

Finally, our measure of block repertoire cannot detect whether a block is used correctly or if a user actually understands the function of the block. This is a challenge with quantitative measures of learning in general, and it has been shown through qualitative work [3] that there are scenarios where a learner uses a given Scratch block by trial and error without necessarily understanding how it works. We present our work in the hope that future research will critique and build upon our approach and measures.

DISCUSSION

This paper's contribution is support for the theory that novice learners have better learning outcomes when learning in their own language. We present models that estimate a positive association between the growth rate of users' repertoires of programming blocks and the translation of their programming language and interface into their local languages. We do not know if these results are generalizable beyond the users and countries in our dataset. We believe that our work points to the possible benefits of supporting localization for designers of educational programming languages and environments.

Our estimated effect size is small and reflects a difference of only several blocks over the full trajectory of some of Scratch's most active users. However, we remain optimistic about these results for two reasons. First, a single block reflects a very large proportion of most users' repertoires. Second, there are several plausible scenarios, discussed above, that might lead to an underestimation of our effect. Of course, as we explain in our limitations, establishing a causal effect remains a goal for future work.

Perhaps more important is the effect that localization has on the degree to which engagement and learning is possible in the first place. After all, Scratch is almost unique among programming languages in that it provides a completely localized interface, including a translated version of the programming language itself. Although we have presented evidence in support of the claim that young programmers learn more quickly using localized interfaces, the most important effect of localization, not captured by our analysis, may be that being able to engage in ones' primary language supports users who would otherwise not learn to code at all.

ACKNOWLEDGMENTS

We are grateful to the Lifelong Kindergarten group at the MIT Media Lab for creating Scratch and for continuing to support the growth of the community and the technical platform. We would also like to acknowledge Mitchel Resnick, Natalie Rusk, Nathan TeBlunthuis, and our anonymous reviewers for their support and thoughtful feedback on this paper. Financial support for this work came from the National Science Foundation (grants DRL-1417663 and DRL-1417952).

REFERENCES

1. Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2015. Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software* 67, 1 (2015), 1–48. DOI:http://dx.doi.org/10.18637/jss.v067.i01

2. Karen Brennan. 2012. ScratchEd: Developing support for educators as designers. *Designing with teachers: Participatory approaches to professional development in education* (2012), 67–77.

3. Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*. AERA, Vancouver, Canada. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

4. Dörthe Bühmann and Barbara Trudell. 2008. *Mother tongue matters: Local language as a key to effective learning*. Technical Report. United Nations Educational, Scientific and Cultural Organization (UNESCO), Paris, France. http://unesdoc.unesco.org/images/0016/001611/161121e.pdf

5. Sayamindu Dasgupta, William Hale, Andrés Monroy-Hernández, and Benjamin Mako Hill. 2016. Remixing As a Pathway to Computational Thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1438–1449. DOI: http://dx.doi.org/10.1145/2818048.2819984

6. Janina Kahn-Horwitz, Joseph Shimron, and Richard L. Sparks. 2006. Weak and strong novice readers of english as a foreign language: Effects of first language and socioeconomic status. *Annals of Dyslexia* 56, 1 (2006), 161–185. DOI: http://dx.doi.org/10.1007/s11881-006-0007-1

7. Luis A. Leiva and Vicent Alabau. 2015. Automatic Internationalization for Just In Time Localization of Web-Based User Interfaces. *ACM Trans. Comput.-Hum. Interact.* 22, 3 (May 2015), 13:1–13:32. DOI: http://dx.doi.org/10.1145/2701422

8. M Paul Lewis, Gary F Simons, and Charles D Fennig (Eds.). 2016. *Ethnologue: Languages of the world (online edition)*. Vol. 19. SIL International, Dallas, TX. http://www.ethnologue.com/

9. Aaron Marcus, Nuray Aykin, Apala Lahiri Chavan, Donald L. Day, Emilie West Gould, Pia Honold, and Masaaki Kurosu. 2000. Cross-cultural User-interface Design: What? So What? Now What?. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems (CHI EA '00)*. ACM, New York, NY, USA, 299–299. DOI:http://dx.doi.org/10.1145/633292.633468

10. MIT Scratch Team. 2013. Scratch Statistics. (2013). https://scratch.mit.edu/statistics/ Accessed: 2016-10-03.

11. Andrés Monroy Hernández. 2007. ScratchR: sharing user-generated programmable media. In *Proceedings of the 6th international conference on Interaction design and children (IDC '07)*. ACM, New York, NY, USA, 167–168. DOI: http://dx.doi.org/10.1145/1297277.1297315

12. Harry Anthony Patrinos and Eduardo Velez. 2009. Costs and benefits of bilingual education in Guatemala: A partial analysis. *International Journal of Educational Development* 29, 6 (2009), 594 – 598. DOI: http://dx.doi.org/10.1016/j.ijedudev.2009.02.001

13. Helen Pinnock and Gowri Vijayakumar. 2009. *Language and Education: The Missing Link: How the Language Used in Schools Threatens the Achievement of Education for All*. CfBT Education Trust, Reading; London. http://www.unesco.org/education/EFAWG2009/LanguageEducation.pdf

14. Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (Nov. 2009), 60–67. DOI:http://dx.doi.org/10.1145/1592761.1592779

15. Patricia Russo and Stephen Boor. 1993. How Fluent is Your Interface?: Designing for International Users. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 342–347. DOI: http://dx.doi.org/10.1145/169059.169274

16. Nakatani Shuyo. 2010. Language detection library for Java. (2010). http://code.google.com/p/language-detection

17. Judith D. Singer and John B. Willett. 2003. *Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence*. Oxford University Press, USA.

18. Jeroen Smits, Janine Huisman, and Karine Kruijff. 2008. Home language and education in the developing world. *Paper commissioned for the EFA Global Monitoring Report 2009, Overcoming Inequality: why governance matters* (2008). http://unesdoc.unesco.org/images/0017/001787/178702e.pdf

19. Judith Uchidiuno, Amy Ogan, Kenneth R. Koedinger, Evelyn Yarzebinski, and Jessica Hammer. 2016. Browser Language Preferences As a Metric for Identifying ESL Speakers in MOOCs. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale (L@S '16)*. ACM, New York, NY, USA, 277–280. DOI: http://dx.doi.org/10.1145/2876034.2893433

20. UNESCO. 1953. *The Use of Vernacular Languages in Education*. Technical Report 8. United Nations Educational, Scientific and Cultural Organization (UNESCO), Paris, France. http://unesdoc.unesco.org/images/0000/000028/002897eb.pdf

21. Benjamin Xie and Hal Abelson. 2016. Skill Progression in MIT App Inventor. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Cambridge, UK. DOI: http://dx.doi.org/10.1109/VLHCC.2016.7739687

22. Seungwon Yang, Carlotta Domeniconi, Matt Revelle, Mack Sweeney, Ben U. Gelman, Chris Beckley, and Aditya Johri. 2015. Uncovering Trajectories of Informal Learning in Large Online Communities of Creators. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale (L@S '15)*. ACM, New York, NY, USA, 131–140. DOI: http://dx.doi.org/10.1145/2724660.2724674