Is Programming Knowledge Related To Age?

An Exploration of Stack Overflow

Patrick Morrison and Emerson Murphy-Hill
Computer Science
North Carolina State University
Raleigh, NC
{pjmorris, emurph3}@ncsu.com

***Abstract*— Becoming an expert at programming is thought to take an estimated 10,000 hours of deliberate practice[1]. But what happens after that? Do programming experts continue to develop, do they plateau, or is there a decline at some point? A diversity of opinion exists on this matter, but many seem to think that aging brings a decline in adoption and absorption of new programming knowledge. We develop several research questions on this theme, and draw on data from StackOverflow to address these questions. The goal of this research is to support career planning and staff development for programmers by identifying age-related trends in StackOverflow data. We observe that programmer reputation scores increase relative to age well into the 50's, that programmers in their 30's tend to focus on fewer areas relative to those younger or older in age, and that there is not a strong correlation between age and scores in specific knowledge areas.**

***Keywords*— Stack Overflow, aging, Social Media, Programming Knowledge, Mining, Software Repositories**

I. INTRODUCTION

Demand for software continues to increase, and demand for programmers shows no sign of flagging. As many existing initiatives consider how to draw more people in to programming, we turn our attention to retaining those already in the field. There is a diversity of opinion on the role of age in programming performance. Comments from a recent thread[2] on the Stack Exchange programmers discussion website illustrate this:

> *"We know that older people have more experience and better judgment, but younger people seem more likely to have used specific technologies we're using, and we like people who can hit the ground running."*
>
> *"There's no inherent "Too Old," just perceptions on the part of the interviewers."*
>
> *"Basically it says that the Valley prefers younger candidates who will put in allnighters for lower wages, and advocates that experienced programmers move into management positions after they hit a certain age."*

We cannot directly assess programming performance, but we can explore programming knowledge. Stack Overflow (SO)[3] is a programming discussion website based on questions and answers about programming that are provided by the online community using the site. As of February 2013, Stack Overflow has over 1.6 million registered users and 4.5 million questions. Registration is not required to use the site, so it is likely that many more users of the site exist than are indicated. SO provides access to its underlying data through online queries[4], and through various forms of data export files.

Over 300,000 SO users have specified their age, which suggests to us that it might prove a suitable candidate for exploring questions regarding age and programming.

In this paper, we use SO data to explore the relationship between aging and programming knowledge. We theorize that age has a positive effect on quality and breadth of programming knowledge, at least up to some point. We also address to what degree older programmers acquire knowledge about newer technologies.

We translate our theory in to several research questions:

- RQ1: Does age have a positive effect on programming knowledge?
- RQ2: Do older programmers possess a wider variety of technologies and skills?
- RQ3: To what degree do older programmers learn new technologies?

II. AGING AND EXPERTISE

Research on the effects of age on cognitive ability suggests that there are tradeoffs in the strengths available over time, among them 'fluid' and 'crystallized' intelligence, where 'fluid' intelligence (Gf) is indicative of the ability to identify complex relations and make inferences based on them, while 'crystallized' intelligence (Gc) 'reflects experience, breadth of knowledge, comprehension, judgment, and wisdom' [1]. Several models exist for how fluid and crystallized intelligence change over time, typically featuring increases in both during early life, and declines in fluid intelligence later in life. Understanding how these factors play out for programmers could assist in staff development and training, resource allocation and individual career planning, if relevant data can be acquired.

[1] http://norvig.com/21-days.html

[2] http://programmers.stackexchange.com/questions/370/how-old-is-too-old

[3] http://stackoverflow.com/

[4] http://data.stackexchange.com/stackoverflow/queries

III. Data collection and analysis

We initially explored the SO data via downloading the 2013 MSR Mining challenge PostgreSQL data dump [2]. We then built equivalent SQL queries to extract data from the live site. The overall user population of SO is 1694981 programmers, with an average age of 30.3, st. dev. 8.2. In order to emphasize the evaluation of knowledge over time, we based our collection and analysis sample of programmers to study on the following criteria:

- Users aged 15 to 70, forming a 'working age' interval, and excluding outlier age values like 99, and Null.
- Users who answered questions in 2012. SO 'Posts', which may be questions or answers, are assigned a creation date, allowing each post to be associated with a year. SO has roughly doubled in use and data quantity annually since its introduction in 2008, with 2012 having the greatest quantity of data. Further, we wanted to avoid rolling more than a single year's worth of data in to a comparison based on age. Finally, we selected users who had answered questions out of a belief that answers are more reflective of programming knowledge than questions. It is possible to ask a good question that cannot be answered, but it seems less likely to have a high-scoring answer that is not understood or wrong.
- Reputation between 2 and 100,000. SO calculates a 'reputation' value for each user[5], reflecting site familiarity, subject expertise and peer respect. For the purposes of our analysis, we treat SO reputation as a proxy for programming knowledge. While neither actual on the job performance, nor a 'work sample' test, the ability to answer questions about programming is frequently used in hiring interviews for programmer positions. We eliminated 1, which is the default reputation assigned at user creation, and dropped reputations over 100,000 as outliers.

The resulting sample contains 84,284 users, with a mean age of 29.02, st. dev. 7.0, and a mean reputation of 1073.9, st.dev. 3975.2. Fig. 1 presents a breakdown of user count and reputation by age group for the sample.

In the figures we present in this paper, all regression lines are plotted using LOESS smoothing, which uses 'local regression' to model and plot a smoothed line based on response and predictor variable values [3]. We have not proposed or developed statistical models for the processes that may be present, and so the smoothing lines are offered as suggestions rather than as attempts to model or predict. The shade area surrounding the line indicates the confidence interval calculated by the smoothing algorithm.

Over the course of this investigation, we used PostgreSQL, The R Programming Language, Microsoft SQL Server and Excel and the facilities provided by SO and the StackExchange web site network. R packages used include ggplot2 and reshape.

[5] http://meta.stackoverflow.com/questions/7237/how-does-reputation-work

IV. Research methodology and results

A. Does age have a positive effect on programming knowledge?

As discussed above, we treat reputation as a proxy for programming knowledge. We queried the SO data query site according to the criteria specified above, and collected the number of programmers, total reputation, and total number of SO membership months for each age in the range 10 to 70. We normalized total reputation by person months in order to correct for length of membership on SO. We calculate 'membership months' as the number of months between the usr creation date and the day the query was run. As shown in Fig. 1, the number of programmers is roughly normally distributed around age 29, though skewed right. Reputation (Fig. 2) is roughly linearly increasing from age 10 in to the 50's. We ran a linear regression (R, lm(total_reputation/person_months ~ age), which indicated a positive slope of .52 total reputation points/person month per year, p.value .0029 (Pr(>.05), R^2 = .08. This suggests that there is a positive relationship between age and reputation on SO.

B. Do older programmers possess a wider variety of technologies and skills?

We theorize that programmers acquire technology and skill knowledge as they progress in their careers.

SO provides a 'tag' feature, allowing each question asked to be annotated with one or more terms indicating the subject matter of the question, e.g. 'javascript', 'c', 'algorithm', 'design-patterns'. We can trace users to the questions they ask and answer, and to the tags associated with each question. We

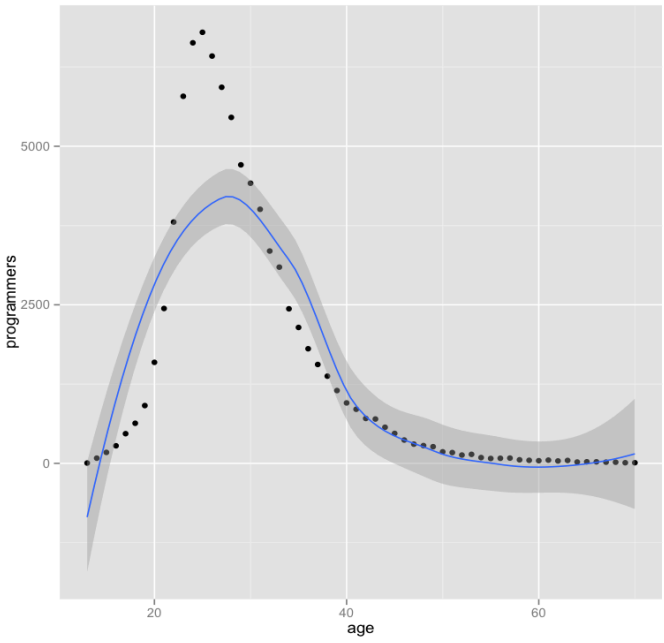
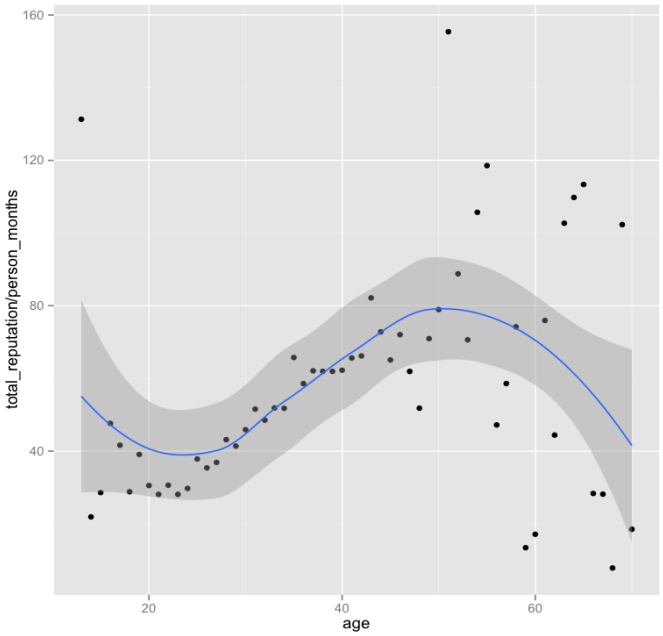


Figure 1: SO user count by age



count the references to each tag, grouped by age and normalize by number of programmers reporting that age. We would expect a plot of this quantity to increase in some proportion to age.

We built an SQL query counting unique tags used by each age, and normalized this total by the number of programmers of that age. A plot of the resulting data, Fig. 3, indicates that our expectations were incorrect; there is initially a decline in the mean number of tags per programmer, bottoming around age 30, followed by an increase in the 40's and 50's and dispersion in the 60's.

C. To what degree do older programmers learn new technologies?

It appears from the preceding analysis that programmers continue to acquire knowledge over time. We conjecture that they acquire knowledge in new technologies, and that this can be measured by considering answer scores. Each answer is assigned a score, based on the number of votes for or against

Figure 2: SO average reputation by age

the answer. If older programmers learn newer technologies, we might expect them to have similar or higher answer scores to younger programmers for these technologies. In order to test this notion, we group programmers as 'older' and 'younger' programmers and apply the one-sided Welch Two-Sample t-test for a set of new technologies. We take older programmers to be those aged greater than one standard deviation above the mean (29+7 = 36), 37 and above. The null hypothesis is that older programmers have the same scores as younger programmers for new technologies. If older programmers have statistically significant low t-values, it would indicate that they do not learn new technologies.

To avoid a temptation to pick examples showing strong age-related performance, we pre-selected the following tags as representative of technologies released within the last 5-10 years: clojure, django, git, ios, jquery, linq, mongodb, ruby-on-rails, silverlight and windows-phone-7.

We built an SQL query to collect all answers for each of these tags according to the user and answer criteria described previously

We then ran t-tests comparing older and younger programmers for each tag. Table 1 presents the results for our pre-selected tags.

There are two tags, 'ios' and 'windows-phone-7', for which there is a statistically significant deviation from the null hypothesis. These may indicate places where knowledge of older technologies, e.g. the 20-year old Objective-C foundation of 'ios', gives older programmers an advantage. Given the strength of the relationship between age and the selected new technologies is relatively weak, we do not have strong evidence against older programmers learning new technologies. It appears that older programmers do learn new technologies.

V. LIMITATIONS

Does the SO population represent the programmer population? US statistics on programmer employment [4] suggest that the age distribution of professional programmers skews older than the user distribution of SO. It is possible that SO represents a kind of 'early adopter' rather than the programming profession. This also has implications for the age data; perhaps younger programmers join as a matter of course, while the older developers that join may only do so if they know themselves to be especially knowledgeable. Determining the relationship between the SO user base and the programmer population is necessary before inferences can be made for prediction, planning or other purposes.

Does SO reputation measure programming knowledge? High reputation scores may reflect efforts in resume building and self-promotion as well as programming knowledge. It is

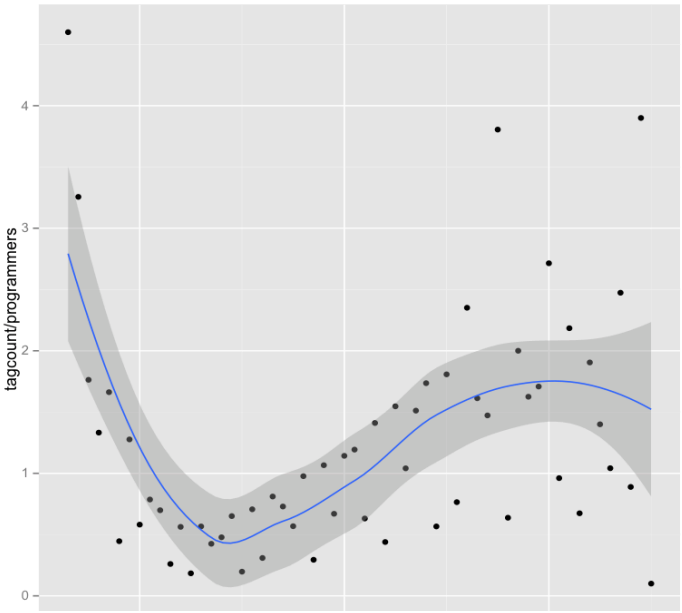


Figure 3: Unique tags by age

Table1: Ten 'new' technologies and younger/older statistical comparisons

tag	t-value	p-value	mean-younger	sd-younger	n-younger	mean-older	sd-older	n-older
clojure	-0.85	0.802	3.25	3.58	539	2.95	2.88	84
django	0.02	0.49	0.91	2.01	4763	0.91	1.72	470
git	0.32	0.62	1.82	3.44	2852	1.78	2.23	426
ios	4.3	8.87E-06	0.96	3.19	19627	1.25	2.9	2119
jquery	0.36	0.358	0.78	2.02	28546	0.79	1.96	3016
linq	0.71	0.24	1.26	2.28	2366	1.34	2.24	431
mongodb	-0.47	0.68	0.95	1.69	2093	0.91	1.38	252
ruby-on-rails	0.52	0.3	0.89	2.45	11089	0.93	2.22	1061
silverlight	0.55	0.29	0.63	1.23	1487	0.69	1.48	235
windows-phone-7	3.41	0.0004	0.68	1.48	2158	1.16	1.68	153

possible that the causation between age and programming knowledge exhibited in the data is because higher-knowledge individuals choose to stay active and engaged later in life, rather than because individuals gain knowledge over time, a point made by Hultsch et al [5].

Does measuring programming knowledge say anything about programming ability? As previously discussed, answering knowledge questions has been used to inform programmer hiring decisions almost universally, but the results of depending only on technical interviews are not always satisfactory. High SO question and reputation scores may indicate a talent for explanation and for clever writing more than an ability to translate knowledge in to code.

We are not convinced that our means for answering RQ3 is fair, although we do not yet have a better procedure. Restricting the population to only those users who answer questions, measuring scores and normalizing by the number of people answering questions all reduce the amount of support available as evidence for use of new technologies by older programmers, and weaker statistical measures may suit for establishing use, e.g. simple counts of the number of different ages of users asking and answering questions about a given technology/tag.

VI. Conclusions and Future Work

We have shown a correlation between age and SO reputation, which may indicate that programming knowledge can be maintained at a high level in to a person's 50's and 60's. It appears that older SO users not only can acquire additional knowledge, but that they acquire knowledge of new technologies, in the case of the technologies we have examined. Further investigation is needed to determine under what conditions this occurs.

This has implications for staff development and career planning. If the trend shown in the data of programmers leaving the profession before they have fully developed were present in the software development industry, slowing it would produce an effective increase in the number of trained programmers available for developing software. Studying and acting on this information at the organizational level might call for new perspectives on the part of boards, executives, management and human resources in recruiting and promotion. At the individual level, awareness of high performance on the part of others can inspire efforts to continue or to improve.

Research into the nature of the SO population and its relation to the programmer population at large needs to be conducted, in order to support inferences to the programmer population. Further investigation is needed of how SO measures, such as user reputation and question scores translate in to programming knowledge and ability. Research on the relationship between age and knowledge on SO should be related to the existing literatures on aging and on the development of expertise.

Ultimately, all information about personal development should translate in to individual questions, such as 'How do I improve?', 'What do I need to learn?', and 'Who can help me learn it?'. SO already offers a rich resource in this regard, and many opportunities exist to improve its use through research.

Acknowledgements

This paper and its authors owe much to discussions with John Slankas, John Majikes, Jen Davidson, and the entire CSC 791 class of Spring 2013.

References

[1] F. I. M. Craik and E. Bialystok, "Cognition through the lifespan: mechanisms of change," *Trends in Cognitive Sciences*, vol. 10, no. 3, pp. 131 – 138, 2006.

[2] A. Bacchelli, "Mining Challenge 2013: Stack Overflow," in *The 10th Working Conference on Mining Software Repositories*.

[3] W.S. Cleveland and C.L. Loader, "Smoothing by Local Regression: Principles and Methods.," in *Statistical Theory and Computational Aspects of Smoothing*, W. Hardle and M. G. Schmick, Eds. New York: Springer, 1996, pp. 10–49.

[4] *Occupational Outlook Handbook, ch. Computer Software Engineers and Computer Programmers*. US Department of Labor, 2011.

[5] D.F. Hultsch, C. Hertzog, B.J. Small, and R.A. Dixon, "Use it or lose it: engaged lifestyle as a buffer of cognitive decline in aging?," *Psychology of Aging*, vol. 14, no. 2, pp. 245–63, Jun. 1999.