No half-measures: A study of manual and tool-assisted end-user programming tasks in Excel

Rahul Pandita*†, Chris Parnin†, Felienne Hermans‡, Emerson Murphy-Hill†
*Phase Change Software, Golden, CO, USA
†North Carolina State University, Raleigh, NC, USA
‡ Delft University of Technology, Delft, Netherlands
Email: rpandita@phasechange.ai, cjparnin@ncsu.edu, f.f.j.hermans@tudelft.nl, emerson@csc.ncsu.edu

***Abstract*—The popularity of end-user programming has lead to diverse end-user development environments. Despite accurate and efficient tools available in such environments, end-user programmers often manually complete tasks. What are the consequences of rejecting these tools? In this paper, we answer this question by studying end-user programmers completing four tasks with and without tools. In analyzing 111 solutions to each of these tasks, we observe that neither tool use nor tool rejection was consistently more accurate or efficient. In some cases, tool users took nearly twice as long to solve problems and over-relied on tools, causing errors in 95% of solutions. Compared to manual task completion, the primary benefit of tool use was narrowing the kinds of errors that users made. We also observed that partial tool use can be worse than no tool use at all.**

I. Introduction

End-user programming [10] environments are designed to empower people without the significant knowledge of programming languages to efficiently perform tasks. These people can take advantage of automated procedures to solve problems that might otherwise have to be performed manually. Such tools come in many forms: as shortcuts and macros in editors, as stand-alone command-line programs, or formula operations in spreadsheets. Both research and practice suggest that certain tools can improve software quality and reduce development time (for example, Ko and Myers' Whyline [24]).

Despite the availability of tools and evidence that they can help, even professional developers oftentimes perform tasks manually instead of leveraging a tool. For instance, Murphy-Hill and colleagues report that programmers performed 90% of refactorings manually, despite having refactoring tools easily available [30]. Data scientists often adopt workflows that involve many manual steps when performing tasks such as data collection, data cleaning, and analysis [17]. However, manual task completion is not without consequences; for instance, developers make more errors when refactoring manually [15]. In contrast, improper tool configuration can also cause errors. For instance, unintended formatting of gene data in Excel has lead to widespread error in many scientific publications [36].

How do developers decide whether to use a tool or perform the task manually? Tversky and colleagues [34], [33] theorized that unlike automated systems that arrive at a decision based on an objective measure of probabilities derived by compounding individual simple probabilities, humans decisions are based on simple heuristics. These heuristics may interfere with a developer's ability to accurately estimate the payoff in using a tool versus the risk of introducing manual errors. Similarly, Blackwell and Green's attention investment model [7], [6] explains the decision to invest in learning a new tool or skill based on several factors such as risk and expected payoff. These theories suggest that end-user programmers may make poor decisions about the risk of performing manual work and the perceived effort and benefit in learning to use a tool.

In this paper, we study ***what contributes to programmers' decision to do manual work when the option to automate exists***. We enlisted online participants for four data extraction and data calculation tasks, and analyzed 111 responses for each of the tasks. Our main contribution is a study that explores the effectiveness and efficiency of end-user programming task performance with varying levels of automation.

From the study, we observed that neither complete automation nor a manual approach was consistently more accurate or efficient. We also observed that partial automation is sometimes worse than a manual approach. Additionally, many participants reported that although manual approach was not ideal they did not know what (and how) tools to leverage for automation. We found that most of the behavior could be explained by people either underestimating or overestimating factors related to risk and configuration effort in using tools. We recommend several design guidelines that improves the ability for users to estimate the effort involved in finding and learning tools for solving a given problem.

II. Study Design

We analyzed tasks performed in Microsoft Excel because it is one of the most widely used programming environments, with 1.2 billion users [3]. Furthermore, Excel users have access to a wide spectrum of tools or automation options, from traditional programming in the form of Visual Basic for Applications (VBA) to Excel formulas to commands like filtering and sorting. For this study, we consider use of formulae, macros, and menu functions in Excel as tool use.

Our study seeks to answer the following research questions:

1) How did end-users solve the tasks?
2) How effective is manual effort versus automation, in terms of time and error rates?
3) What factors influence the choice of problem-solving strategy?

978-1-5386-4235-1/18/$31.00

A. Participants

We recruited participants via Amazon Mechanical Turk (MTurk) [1], an on-line crowd-sourcing marketplace to facilitate and coordinate human intelligence tasks (HITs). Recent research shows that MTurk is an appropriate place to recruit study participants for behavioral research [28]. We recruited participants who have completed at least 1000 prior HITs as vetting criteria for reliability and experience of the participant. We paid participants $3 USD for completing a set of tasks. We required participants to have Excel installed on their system.

B. Tasks

We designed tasks that: can be done in a variety of ways, including manually; cannot be completely solved with one tool; and reflect common tasks, namely *reading and extracting*, and *searching and filtering* [31]. We designed two tasks (1 and 2), each with two sub-tasks (A and B), and did not control for the order in which participants performed sub-tasks:

1) Task 1A: This sub-task required participants to extract zip codes from 61 lines of text, each line containing a postal address. Zip codes appeared strictly at the end of the text. A typical US zip code is 6 digits long. However, to break regularity, two zip codes contained 9 digits and one address contained a Canadian alphanumeric zip code. Figure 1a shows a list of addresses and extracted zip codes.

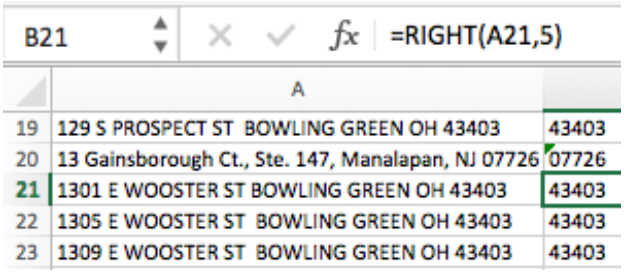
2) Task 1B: This sub-task required participants to extract zip-codes from 40 lines of text. The text in each line was a concatenation of name, postal address, phone number, and email. In this sub-task the zip code appeared interleaved in the text instead of at the end.

3) Task 2A: This sub-task presented participants with a list of words consisting of names of 229 fruits and vegetables (Figure 1c). We asked participants to count the number of words starting with "A", starting with "B", starting with "P" (deliberately not "C" to break regularity), and finally count the words containing "berries".

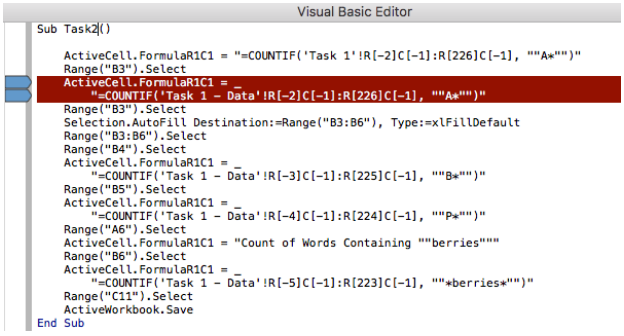
4) Task 2B: This sub-task presented participants with a list of 1011 numbers, each representing the average number of hours a person sleeps. We asked participants to count the number of people that sleep 3–5 hours, that sleep 6–7 hours, and that sleep 8–9 hours.

C. Procedure

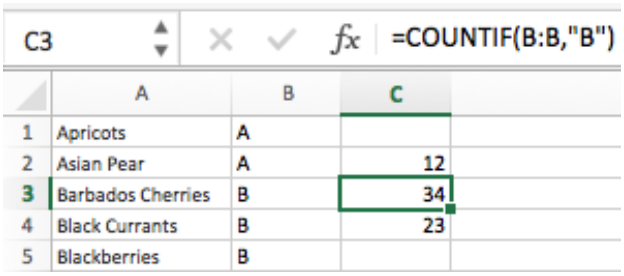
In a survey, participants were asked about their familiarity with Excel, from "Not at all familiar" to "Extremely familiar." To help analyze how participants completed the tasks, we instructed participants to record macros, which recorded all Excel actions performed by participants. Participants could choose any strategy to solve the tasks. We also instructed participants to record the time spent on each sub-task, since macros do not capture timing. We provided participants with an Excel file containing a task to be performed. Although participants were free to finish the task at their own pace, they had to submit their solution within one hour to qualify for the compensation. Finally, we provided participants with a



(a) Task 1A: Extract zip codes from address strings.



(b) A recorded macro describing a participant's task solution.



(c) Task 2A: Counting prefixes and substrings in words.

Fig. 1: Screenshots of Tasks and Macro in Excel

link to upload their completed Excel files. They also answered the following questions about each sub-task:

1) About how long did the sub-task take you?
2) How did you approach the sub-task? Which tools, features, or functions did you use to help?
3) Do you think this is the most efficient strategy to solve the sub-task? If not, what prevented you from using a more efficient strategy?
4) How might you change your strategy if there were many more items to process?
5) Did you search online for help for this sub-task? What phrases did you search for?

D. Analysis

1) Cleaning Data: We first removed incomplete data. 254 people attempted Task 1 and 260 attempted Task 2. After excluding data where participants did not record macros, we selected 111 participants for each task. We did not necessar-

ily have 222 participants, since some participants may have independently completed both tasks.

2) Identifying Strategies: We analyzed the submitted Excel files and macros to extract the following information: the tools (functions or commands) used by the participant; the number of correctly answered questions; and any mistakes the participants made. In Figure 1b, we display an example task recording. We also recorded the self-reported time spent by participants for each sub-task. Finally, we classified each attempted sub-task into one of the following strategies:

- **Manual**: The participant does not use any tool.
- **Fully-automated**: The participant exclusively uses tools.
- **Semi-automated**: The participant uses a mix of the above strategies.

III. Results

A. How did people solve the tasks?

Overall, we were surprised with the variety and creativity of solutions that participants used. Some participants wrote VBA scripts. Others used semi-automated techniques, such as first sorting numbers, and then manually selecting rows to get a count. A few participants used creative solutions, such as using the find-and-replace command for Task 2A, then using the resulting popup box that tells you the numbers of items that were replaced. Many participants used formulas to calculate solutions. For example, for Task 1A, it was popular to use a function like `RIGHT` to extract the last 5 digits to get the zipcode. For Task 2B, it was popular to use `COUNTIF` to count the number of items that met the search criteria.

A description of all the strategies that participants used and their frequency of use is on FigShare [2]. The number of solutions for each task ranged 5–8 For instance, one participant describes his approach for Task 1B as:

> *Again examined the data to look for a pattern I could use to extract the data. Used the Formula ribbon descriptions to get the proper search function (ie, find, search or lookup). Used `MID` and `SEARCH` to retrieve the 5 digit zip code. Visually inspected the results and found that my original pattern (search for " ???-") did not work for one record. Changed the pattern for the search to get the desired result.*

Grouping these strategies by level of tool use, we can see in Table I how often participants decided to manually perform a task or use a tool instead. The use of manual or tool-assisted strategies greatly varied with the task performed. Many participants performed the zipcode extraction task manually (42.3% for Task 1A and 53.2% for Task 1B). In contrast, very few participants performed the counting tasks manually (4.5% for Task 2A and 3.6% for Task 2B). Instead, participants performed the counting tasks in a semi-automated manner (56.8% for Task 2A and 55.9% for Task 2B), using a tool to start but then finishing the calculation manually.

While analyzing macros we observed that participants spent a significant fraction of effort on browsing and getting familiar with the data. We computed the effort spent on browsing by counting the fraction of macro statements that correspond to browsing (scrolling and selection) in Excel. For scrolling, we identified macro statements starting with : 1) "ActiveWindow.Scroll", 2) "ActiveWindow.SmallScroll" or 3) "ActiveWindow.LargeScroll". For selection, we identified macro statements ending with ".Select" such as `Range("D230").Select`

TABLE I: Manual and tool-assisted strategy usage rates

Task	Automated (%)	Semi (%)	Manual(%)
Task 1A	23 (20.7%)	41 (36.9%)	47 (42.3%)
Task 1B	25 (22.5%)	27 (24.3%)	59 (53.2%)
Task 2A	43 (38.7%)	63 (56.8%)	5 (04.5%)
Task 2B	45 (40.5%)	62 (55.9%)	4 (03.6%)

On average 65% statements corresponded to browsing data. We also observe that the fraction of browsing statements depend on task, strategy, and correctness ($p < .001, \chi^2$). Participants browsed more in Task 2 than in Task 1; this finding can be explained by the larger amount of data that participants needed to process in Task 2. Surprisingly, participants who adopted a manual strategy for the tasks had *fewer* browsing statements than participants who used tools; we hypothesize that this may be because participants needed to inspect the data to understand its structure before applying a tool and also needed to inspect the data after the tool was applied to ensure correctness. Finally, participants who performed the task correctly tended to browse more; this could be explained by them taking extra care in reviewing their results.

> Participants completed tasks in a variety of ways, including by repurposing tools. Furthermore, participants spent significant effort on browsing data, which correlated with task, strategy, and correctness.

B. Automated vs. manual performance

1) How fast were people at solving the task?: Mean times for all participants and strategies are presented in Figure 2. The length of each bar represents the time spent on a task using a strategy. One overall conclusion is that there is no consistently superior strategy. For example, on average to perform Task 1B, participants spent 412 seconds to do so manually, 564 seconds semi-automatically, and 790 seconds fully automatically manner. However, for Task 1A, we participants took 360 seconds to perform the task in automated manner, 416 seconds manually, and 564 seconds semi-automatically. Finally, although we report manual times for participants that did Task 2A ($n = 5$) and Task 2B ($n = 4$), these are primarily outliers who submitted completely incorrect answers.

Interestingly, the slowest performers in all tasks used tools exclusively to solve tasks. There were a handful of individuals who used fully automated solutions to solve the problem quickly, but the performance gain was only moderate over other manual users who were almost as fast. Why were tool users often so slow? According to self-reports in the post-survey, participants spent significant time trying to understand

TABLE II: Accuracy rates by strategy and task.

Task	Automated (%)	Semi (%)	Manual (%)
Task 1A	4.4%	31.7%	44.7%
Task 1B	88.0%	40.7%	44.0%
Task 2A	53.4%	34.9%	40.0%
Task 2B	71.1%	69.4%	25.0%

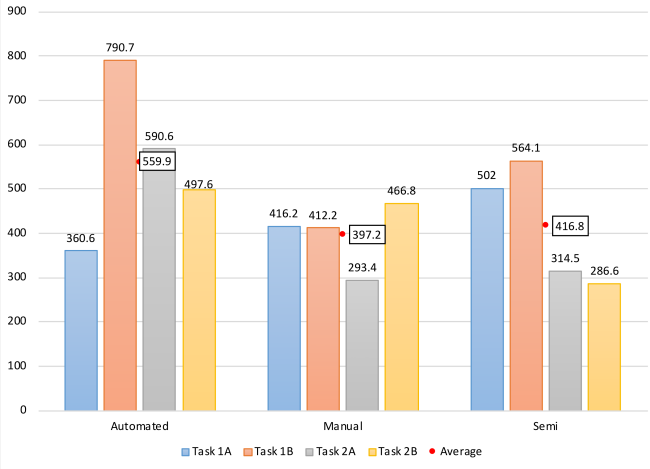


Fig. 2: Average time (in seconds) versus strategy. The • indicates the average time across all tasks.

or adapt the tool to the problem. Some slow performance in manual and semi-automated approaches could be explained by participants that attempted a more automated solution, but then abandoned their approach and completed the task in a manual fashion. One participant describes this situation:

> *Task 2A: Seemed straighforward, thought about trying a couple things, then figured I was overthinking it. I just did a sort and then highlighted the items and looked at the count.*

There was no consistently fastest strategy, but tool-only users were often the slowest performers.

2) How correct were people in solving the task?: For this analysis, we measured correctness as the percentage of correct answers provided for a sub-task. We observed that, although tools are generally designed to reduce errors, participants who exclusively used tools were not immune to errors. In fact, many times participants were not only slower using tools, but wrong as well. For example, almost every participant using a tool (96%) made errors for Task 1A. Semi-automated approaches did not fare much better; they had the lowest accuracy ratings for three of the tasks.

Table II displays the accuracy of each strategies for each task. Task 1 has statistically significant differences in accuracy rates by strategy ($p < .01, \chi^2$), but not for Task 2. In Task 1A, automated solutions achieve a low accuracy rate of 4.4%, whereas in Task 1B, they have twice the accuracy rate over manual or semi-automated strategies.

Table III displays a breakdown of speed, accuracy, and strategy. We used any task completion speed below the first percentile for a rating of *slow*, and any task completion speed above the third percentile for a rating of *fast*. If a participant made no errors, we indicate this as *correct*, otherwise, if they made any error, we indicate this as *error*. From this data, we can observe that it was not typical to have a fast solution and be correct. Unfortunately, participants that try to automate their solutions with tools are often still slow and incorrect. Further, no strategy consistently ensured both speed and correctness.

We also analyzed the correctness of task against the participant familiarity with Excel. Figure 3 plots the correctness against familiarity for each strategy. We next plot a regression line for each strategy using LOESS smoothing [11]. Based on LOESS analysis, using a fully-automated strategy results higher correctness across all familiarity levels. Another interesting trend is, while correctness in manual strategy increases as the familiarity increases, there is slight decline in overall correctness as familiarity increase for fully-automated and semi-automated strategies. This slight decline in overall correctness in fully and semi automated strategies may be attributed to blindspots introduced by tool use and familiarity.

Why did participants make so many errors with tool-assisted approaches, especially for Task 1A? One major reason was that participants may have failed to exercise any oversight. While most zip codes were 5 digits, a couple were 9 digits and one was alphanumeric. If participants assumed that all the zip-codes were 5 digit numbers and overlooked exceptions, they can very easily make this mistake. Given that many participants went on to do Task 1B, correctly with a tool, we suspect this is the case. For participants that did notice an error, this was often a reason to switch from a fully automated solution to a semi-automated solution. For example:

> *This one was fairly easy, just needed to lookup right truncation. Of course I overlooked the plus four zips. But there were so few, I just corrected by hand. Still only took 8 minutes.*

Although tools could help improve correctness, they could also introduce blindspots that contribute to devasting error rates. No strategy was consistently accurate.

3) What type of errors did people make?: We wanted to understand the variety of errors that people may make and relate them to different strategy usage. We expected the error categories to be unique to the strategy taken by participants. For each participant, we classified the error they made into a pool of error categories by manually inspecting the recorded macro. From this inspection, we were able to infer the error that participants made in their solutions.

After examining the category of errors participants made, the most clear result was that participants who exclusively used tools made the fewest kinds of errors. That is, although automated users still were inaccurate, the error they made was typically isolated to one specific class of errors, whereas participants who made use of manual or semi-automated solutions had a much wider set of errors made. For example,

TABLE III: Strategy, speed, and accuracy
●(Automated) ◓(Semi) ○(Manual)

	Slow	Moderate	Fast
Task 1A			
Correct	◓◓◓◓◓◓○○○○○○○	●◓◓◓◓◓◓○○○○○○○○○○○○○	◓○
Error	●●●●◓◓◓◓◓◓○○○○○○○○	●●●●●●●●◓◓◓◓◓◓◓◓◓◓○○○○○○○○○○○○○	●●●●●●●●◓◓◓◓◓◓◓◓◓◓◓◓○○○○○○○
Task 1B			
Correct	●●●●●●●●●◓◓◓○○○○○	●●●●●●●●●◓◓◓◓◓○○○○○○○○○○○○○○○○○	●●●●●◓◓◓○○○○
Error	●●◓◓◓◓○○○○○	◓◓◓◓◓◓◓◓○○○○○○○○○○○○○○○	●◓◓◓○○○○○○○○○○○○○○
Task 2A			
Correct	●●●●●●●◓◓	●●●●●●●●●●◓◓◓◓◓◓◓◓◓◓◓◓◓○	●●●●●◓◓◓◓◓○
Error	●●●●●●●●●●●●●◓◓◓◓○	●●◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓○○	●●◓◓◓◓◓◓◓◓◓◓◓◓◓
Task 2B			
Correct	●●●●●●●●●●●◓◓○	●●●●●●●●●●●●◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓	●●●●●◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓◓
Error	●●●●●●◓◓◓◓◓○○	●●●◓◓◓◓◓◓◓◓◓◓○	●●◓◓◓◓

Fig. 3: Correctness vs. Familiarity vs. Strategy

some participants who made use of sorting and manual count strategy for Task 2A had a phonetic sort that caused an error in how items were grouped that was not present for other users. Other users who used the semi-automated approaches often experienced off-by-one errors when they selected data to copy. Finally, users who employed manual strategies often experienced mechanical errors when typing out answers.

Table IV lists the distributions of errors across various categories. We next describe these categories next:

• **Manual**: This category of error includes logical human errors. In Task 1, this category of error included cases where participants did not account for the exception cases (either 9-digit or Canadian Zip code). It is surprising to see 9 participants who performed the task manually also got confused with the exception cases. In Task 2 this category included cases where participants counted incorrect data. For instance, in Task 2A first two questions required participants to count words beginning with "A" and "B". However, the third question required participants to count the words beginning with "P" to break the regularity. We observed that participants counted words beginning with "B" instead.

• **Typing**: This category involved participants making mistake while typing the answers. We observed this class of errors exclusively in Task 1 and was dominated by strategies with some form of manual steps. We suspect the data cleaning nature of task and more participants following manual strategy in Task 1A is reason that this category exclusive to Task 1. The only exception is one participant who typed the formula in the incorrect location for Task 1A.

• **Copy**: Participants often preferred to perform the computation in separate area and then copy the final solution to the designated cells. This category includes participants who forgot to copy the solution.

• **Partial**: This category constitutes instances where participants did not complete the task. This category was exclusive to Task 1. We suspect that participants estimated that Task 1 entailed substantially more work, compared to Task 2. Task 1 involved participants having to extract 61 data elements for Task A and 20 for Task B, whereas, Task 2 involved answering just 4 questions (Task A) and 3 questions (Task B).

• **No Task**: This category constitutes instances where participants did not perform one sub-task at all. We did not analyze the results if participants did not perform both of the sub-tasks.

• **Sort**: This category was exclusive to Task 1 where some participants sorted the data rows before or after extracting the data. This affected the order of the desired output. We are unclear why participants performed the sort.

• **System**: This category was observed exclusively for Task 2, where the functionality of the environment worked in an unexpected way, such as the "phonetic" sort described earlier. In another case, the input data was likely corrupted by the environment for a participant.

Tools reduced the class of errors participants made.

C. *Why did a person perform a task manually?*

We next examine several factors that influence why a participant did not use a tool to perform a computational task.

1) How did the person decide to do the task?: We were interested in understanding the motivations of people in considering (or ignoring) tools for a task. For each participant,

TABLE IV: Error Categories

Task	Str.	Error Category						
		Manual	Typing	Copy	Partial	No Task	Sort	System
1	Man	9	28	17	14	4	1	0
	Semi	11	15	12	10	7	2	0
	Auto	18	1	1	2	4	1	0
2	Man	5	0	1	0	0	0	0
	Semi	54	0	4	0	0	0	3
	Auto	23	0	7	0	2	0	0

TABLE V: Participants responses for "Do you think this is the most efficient strategy to solve the task?"

Task	Response	Automated	Semi	Manual
Task 1	Yes	28	24	49
	Maybe	6	10	4
	No	14	33	42
Task 2	Yes	61	78	3
	Maybe	7	16	2
	No	20	23	4
Total		136	184	104

we systematically went through the post-survey responses. In particular, we analyzed participant responses for question: *"Do you think this is the most efficient strategy to solve the task? If not, what prevented you from using a more efficient strategy?"*

If the participant indicated that their approach was the most efficient for a subtask we classified the response as *Yes*, if not we classified the response as *No*. If the participants response indicated that (s)he was not sure we classified the response as *Maybe*. We did not consider 20 instances of empty or non-applicable responses across subs-tasks.

Table V presents the distribution of responses across Tasks for automated, semi, and manual approaches. Overall, 54% of participants responded *Yes*, 31% responded *No*, 10% responded *Maybe*, and 5% responded *NA* or did not respond. In all, 65% (89/136) of people that followed an automated strategy responded that their strategy was efficient. We were surprised that roughly half (52/104) the people that attempted the task manually also felt their strategy was efficient as well. The responses alluded to the fact that participants felt that the manual strategy was efficient because, it was simple and fast for the small dataset in the tasks, as captured in this response by a participant *"I think this is a pretty quick and dirty way of accomplishing the goal."*

We explicitly asked participants to document what prevented them from using an efficient strategy, if they thought their strategy was not efficient. Two authors independently coded a random subset (10%) of responses. Authors followed the guidelines of open card sort [12], where they created categories based on the data itself. We then compared the results and documented that the authors were in agreement for 77.3% of their classifications. Based on the discussions, the first author then coded the rest of the responses. No new category emerged as the first author coded rest of the responses.

We next list the categories that emerged from the participant

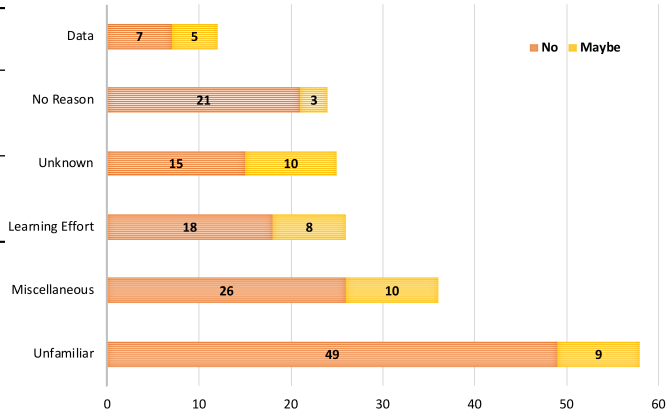


Fig. 4: Reasons for not selecting the most efficient strategy

responses to the question *"If not, what prevented you from using a more efficient strategy?"*:

- **Unknown**: Did not know a better way to do a task, e.g. *"no. I can't think of any other strategy"*
- **Unfamiliar**: The user is cognizant of the existence of tools but was not sure on what (or how) to use them, e.g. *"I am sure there was a formula that would have been faster, but I didn't know it."*
- **Learning Effort**: The user was impeded by perceived effort in learning to use tool, e.g. *"could not figure out how to make it work"*.
- **Data**: The choice was dependent on data, e.g. *"For this set of data I think it was the fastest way to get it done."*
- **Miscellaneous**: This was a catch-all category for responses that did not fit anywhere else. These included reasons such as: tool that did not did not work as expected, participant ran out of time, or participant made incorrect assumptions about the task.
- **No Reason**: Did not provide any reason or the provided reason is vague.

Figure 4 presents our findings on the impediments users face in employing what they consider as an efficient strategy. Most of the users were cognizant of existence of a tool that would allow them to perform the task better (Category "Unfamiliar"). However, they were not sure of either what tool to use or how to use a tool. Not considering the catch-all "Miscellaneous" category, the second most common impediment faced by participants was that they perceived the investment in learning about the tool too high for them to leverage tools in their task. Next participants reported that they were unaware of any other way of accomplishing the task. Followed by participants who thought their approach was not optimal, but data-set forced them to use the approach they chose.

While we anticipated participants to cite *unfamiliarity with tools* and *not knowing any other ways to solve a task* as the reasons for not using an optimal strategy, the explicit "learning effort" category provides opportunity for toolsmiths to design better tools that users perceive as easy to learn.

Users can be dissuaded from tool use based on the perceived learning effort.

IV. Discussion

With the estimate that end-user developers outnumber professional developers by 50 million to 3 million [10], the goal of this research was to gain a deeper understanding of the decision process that end-user developers employ when deciding between manual effort or tool use. Additionally, some of our findings may generalize to professionals developers as well. This section discusses implications of our findings and threats to validity. We first summarise some of the general patterns we observed in the participant behavior.

A. Findings

1) No half-measures: We were surprised to find that participants performing the task either manually or in a complete automated fashion consistently outperformed participants employing a semi-automated approach, in dimensions of task correctness and speed. In general, a participant performing manual actions in the task is at a higher risk of introducing a mechanical errors [4]. However, we suspect that participants who performed the tasks manually were generally more careful to look for and avoid such errors. In contrast, participants performing the task in a semi-automated fashion may not have accounted for the risk of mechanical errors due to manual part of the strategy.

2) Tools reduce the kinds of errors made: We observed that the use of tools not only diminished the number of errors, they also helped participants avoid certain classes of errors. For instance, copy-paste and typing related errors are almost exclusively observed in the cases where participants attempted to perform the task manually or semi-automatedly. Such simple coding mistakes often produce notoriously difficult-to-find defects [20], [21].

Although use of tools did help participants avoid errors in general, Task 1A was an exception. Specifically, users of `RIGHT` function often did not correct for the interleaved exceptions (9 digit and Canadian zip codes). In this case, use of tool may have caused participants to get false sense of correctness by working 58 out of 61 cases and leaving out 3.

3) Large investments can go bust: From our analysis of strategy vs. time and correctness we observed that although some benefited from their investment to use tool, many often spent a long time learning how to get a tool to work and never received the expected payoff (either performed task very slow compared to others, or still made errors).

Further, not all solutions translated well from the first part of the task to the second part of the task. For example, 41 participants attempted Task 1A in using `RIGHT` function which does not lend itself to the Task 1B. In contrast, participants that used the `MID` function, were better able to adapt between tasks. When an investment went bust, participants would often just switch to a manual approach:

> *I couldn't find any way of easily doing the second task and as I had already spent so long on the first I just manually copied and pasted everything I could.*

4) Tool selection factors: There is a large body of work in psychology that studies human decision making. For instance, Khaneman in his book "Thinking Fast and Slow" [23] talks about how human decision making is not always objective and is often affected by biases, beliefs, and heuristics. For instance *conjunction fallacy* [34] is phenomena when a person incorrectly assumes that specific conditions are more probable than a generic one. Another line of work that is relevant to this study is the *law of small numbers* [32] a form of *sampling bias*. When sampling, users focus on the little data at hand, while discounting issues which could occur in other data-sets. These effects were clear in our experiment, where participants were confident they were right even when there were better solutions. Concretely, we observed these effects in play when a significant number of participants incorrectly assumed that zip-codes are always 5 digit numbers on the right of the input string in Task 1A.

We also observed the *availability heuristic* [33] in participant behaviour. Availability heuristic causes a person to be more likely to weigh their judgment towards a recent event instead of objectively evaluating the present situation. We observed that most of the participants did not change their strategy for solving each subtask, even though they spent time adapting the strategies to the new subtask.

B. Implications for Design

The findings from the presented study may help with the design of the tools for facilitating better user interaction and engagement. We next outline some recommendations.

1) Provide estimates for learning effort: We observed that a significant number of participants had difficulty in realistically estimating the time and effort required on their part to understand and configure a tool, and whether that would be worth the investment.

Toolsmiths in a programming environment could assist their users to make better estimates. For instance, Viriyakattiyaporn and Murphy [35] proposed an approach to leverage a programmers history of tool use to actively recommend tool in current context. Likewise, Johnson and colleagues [22] propose leveraging developer knowledge to tailor a tools' notifications.

An estimate of *difficultly* or *time-commitment* of using a tool can further enhance these approaches. For example, a naïve yet effective approach could be to provide an estimate of the time to configure a tool correctly based on how long other (first-time) users took to configure it. Furthermore, programming environments could attempt to actively guess what task users are attempting and provide them with contextual data.

2) Highlight unusual values after applying functions: We also observed that a significant number of participants in Task 1A incorrectly extracted the Canadian and the 9-digit zip codes. Partly because these participants approached the problem using the `RIGHT` function to extract the 5 characters

towards the right of the input string. While this approach worked well for 58 out of 61 cases, the participants still had to manually correct the one case involving the Canadian and two cases involving the 9-digit zip codes. Oftentimes, participants overlooked these exception cases and incorrectly reported the tool output as the correct zip-code.

While a manual inspection to verify tool output is highly recommended, we suggest tools should be preemptive in reminding developers to perform review. We also recommend toolsmiths to design tools cognizant of the "unusual" results to help ease the process of review. For instance, existing body of research on code-smells [13], [18], [19] can be extended to detect and report such instances to the user.

3) Tool Recommendation and Strategy Sharing: In the post-survey, when participants were asked to state the reasons that prevented them from using what they thought was an optimal strategy, 35 responses alluded to the fact participants were unaware of any other ways to solve the task at hand, despite a variety of strategies employed by other participants. Existing program synthesis approaches like FlashFill [16] in part alleviate the problem by automatically proposing a solution as a function of input output relationships. The programming environment can further help such users by recommending alternate strategies based on the current context of the user by leveraging the strategies employed by other users in similar contexts. For instance, environments can leverage concepts from "Programming by example" [26] where a software agent records the activities of users to reproduce them later. As the diversity of such recording grow overtime, these recordings can be queried as a shared resources (online or offline) for alternate strategy recommendation [29].

C. Threats to Validity

The primary threat to external validity is the representativeness of our data and tasks to the real world data cleaning workflows. To address this threat we focus on data extraction and data calculation tasks in Excel, which are typical computational tasks in programming and related fields such as data science. In 2014, New York Times reported [27] that analysts spend up to 80% of their time in cleaning data.

Threats to internal validity include the correctness of the identified functions used by the participants. Since authors manually identified the functions used by the participants, human error may affect our results. To minimize the effect, the authors checked the identified functions against the macros that were recorded by the participants while they were performing the tasks. Additionally, the authors manually coded the survey responses to identify the impediments faced by participants in using tools. To minimize this threat, we followed the safeguards for conducting empirical research proposed by Li [25]. To ensure researcher agreement about the findings, two authors independently analyzed the a random subset of participant responses to identify impediment categories.

These threats could be further minimized by evaluating more tasks in other developer programming environments and different settings. We plan to share various materials on the project web [2], to enable other researchers to emulate our methods to repeat, refute, or improve our results.

V. Related Work

Burnett and colleagues studied the introduction of a new tool for spreadsheets (assertions) that could be used to improve the detection of faults when compared to manual inspection [9]. Likewise, Cunha and colleagues demonstrate that Excel tools based on high level domain models help in avoiding errors [14]. In contrast, Murphy-Hill and colleagues demonstrate that users often do not use a specific set of tools to perform a specialized task (refactoring) [30]. However, we differ from this research in two different ways: 1) Instead of participants being instructed to use a specific tool, participants are given free reign to choose how they solve a task. This allows us to observe of this decision process. 2) Our tasks do not necessarily have a ready-made tool for directly solving the task (unlike performing an extract method refactoring with an extract method tool). Instead, participants must select from a federation of tools for solving the tasks, which sometimes involves manual steps in between application of two different tools.

To understand the decisions users make when selecting tools for problem-solving, Blackwell and Green [7] have proposed the investment of attention model. This framework describes four cost-benefit variables: cost, risk, investment, and payoff that help predict a person's willingness to learn a new skill or try a new tool. Blackwell and Burnett [5] have used this to model adoption of a new tool in a spreadsheet. In a similar fashion, Brandt et al. introduce the concept of 'Opportunistic Programming'[8], which describes a class of developers who adopt a minimum learning style and attempt to find online help specific for solving a task. Consistent with this approach, several participants in our study reported watching tutorial videos or reading blog posts in order to learn a strategy for solving the tasks.

VI. Conclusion

In this paper we found that participants performing Excel tasks using tools took more time on average than participants performing the task manually. However, 63% of participants performed the task correctly using automated solutions, compared to 37% participants that chose manual analysis. We found that most of the behavior could be explained by people either underestimating or overestimating factors related to risk and configuration effort in using tools. Environments that assist in estimating these factors may help future programmers make better choices when it comes to deciding whether to use tools.

Acknowledgment

This material is based upon work supported with funding from the Laboratory for Analytic Sciences and the Science of Security Lablet. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any entity of the United States Government.

REFERENCES

[1] Amazon Mechanical Turk. https://www.mturk.com/mturk/welcome.

[2] Project Website. https://figshare.com/projects/Excel_Study/36959.

[3] Microsoft by the numbers, November 2014. https://news.microsoft.com/bythenumbers/ms_numbers.pdf.

[4] B. Bishop and K. McDaid. An empirical study of end-user behaviour in spreadsheet error detection & correction. *arXiv preprint arXiv:0802.3479*, 2008.

[5] A. Blackwell and M. Burnett. Applying attention investment to end-user programming. In *Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC'02)*, pages 28–. IEEE Computer Society, 2002.

[6] A. F. Blackwell. First steps in programming: A rationale for attention investment models. In *Proceedings of the IEEE Symposia on Human Centric Computing Languages and Environments*, pages 2–10. IEEE, 2002.

[7] A. F. Blackwell and T. R. Green. Investment of attention as an analytic approach to cognitive dimensions. In *Collected Papers of the 11th Annual Workshop of the Psychology of Programming Interest Group (PPIG-11)*, pages 24–35, 1999.

[8] J. Brandt, P. J. Guo, J. Lewenstein, and S. R. Klemmer. Opportunistic programming: How rapid ideation and prototyping occur in practice. In *Proceedings of the 4th International Workshop on End-user Software Engineering*, pages 1–5. ACM, 2008.

[9] M. Burnett, C. Cook, O. Pendse, G. Rothermel, J. Summet, and C. Wallace. End-user software engineering with assertions in the spreadsheet paradigm. In *Proceedings of the 25th international conference on Software engineering*, pages 93–103, 2003.

[10] M. M. Burnett and B. A. Myers. Future of end-user software engineering: beyond the silos. In *Proceedings of the on Future of Software Engineering*, pages 201–211. ACM, 2014.

[11] W. S. Cleveland and S. J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American statistical association*, 83(403):596–610, 1988.

[12] J. Corbin and A. Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.

[13] J. Cunha, J. P. Fernandes, P. Martins, J. Mendes, and J. Saraiva. Smellsheet detective: A tool for detecting bad smells in spreadsheets. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 243–244. IEEE, 2012.

[14] J. Cunha, J. P. Fernandes, J. Mendes, and J. Saraiva. Embedding, evolution, and validation of model-driven spreadsheets. *IEEE Tran. on Software Engineering*, 41(3):241–263, 2015.

[15] X. Ge and E. Murphy-Hill. Manual refactoring changes with automated refactoring validation. In *Proceedings of the 36th International Conference on Software Engineering*, pages 1095–1105, New York, NY, USA, 2014. ACM.

[16] S. Gulwani. Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Notices*, volume 46, pages 317–330. ACM, 2011.

[17] P. Guo. Data science workflow: Overview and challenges' acm blog, 30 october 2013, 2013.

[18] F. Hermans, M. Pinzger, and A. v. Deursen. Detecting and visualizing inter-worksheet smells in spreadsheets. In *Proceedings of the 34th International Conference on Software Engineering*, pages 441–451. IEEE Press, 2012.

[19] F. Hermans, M. Pinzger, and A. van Deursen. Detecting code smells in spreadsheet formulas. In *Proceedings of the 2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 409–418. IEEE, 2012.

[20] W. S. Humphrey. *A discipline for software engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[21] W. S. Humphrey. The personal software process (PSP). 2000.

[22] B. Johnson, R. Pandita, E. Murphy-Hill, and S. Heckman. Bespoke tools: adapted to the concepts developers know. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 878–881. ACM, 2015.

[23] D. Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[24] A. J. Ko and B. A. Myers. Designing the whyline: a debugging interface for asking questions about program behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 151–158. ACM, 2004.

[25] D. Li. Trustworthiness of think-aloud protocols in the study of translation processes. *International Journal of Applied Linguistics*, 14(3):301–313, 2004.

[26] H. Lieberman. *Your wish is my command: Programming by example*. Morgan Kaufmann, 2001.

[27] S. Lohr. NY Times - For big-data scientists,'janitor work' is key hurdle to insights, 2014.

[28] W. Mason and S. Suri. Conducting behavioral research on amazons mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.

[29] E. Murphy-Hill. Continuous social screencasting to facilitate software tool discovery. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1317–1320. IEEE Press, 2012.

[30] E. Murphy-Hill, C. Parnin, and A. P. Black. How we refactor, and how we know it. *IEEE Transactions on Software Engineering*, 38(1):5–18, 2012.

[31] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, volume 5, pages 2–4, 2005.

[32] A. Tversky and D. Kahneman. Belief in the law of small numbers. *Psychological bulletin*, 76(2):105, 1971.

[33] A. Tversky and D. Kahneman. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5(2):207–232, 1973.

[34] A. Tversky and D. Kahneman. Judgment under Uncertainty: Heuristics and Biases. In *Utility, Probability, and Human Decision Making*, pages 141–162. Springer Netherlands, 1975.

[35] P. Viriyakattiyaporn and G. C. Murphy. Improving program navigation with an active help system. In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, pages 27–41. IBM Corp., 2010.

[36] M. Ziemann, Y. Eren, and A. El-Osta. Gene name errors are widespread in the scientific literature. *Genome Biology*, 17(1):177, 2016.