Dylan Rohan a1844790

Analysis of a Fake Controlled Study

Introduction

The purpose of this experiment is to answer a vague descriptive-comparative question, *"which programming language makes developers more productive, JavaScript or java?"*. The researchers appear to have a pragmatic approach, taking the simplest indicators and making approximations of the real world. The experiment outlined would benefit from a more constructivist school of thought. This would encourage more specific word choices and perhaps more valid conclusions.

Essentially, the experiment was going to use JavaScript students to determine whether developers would be 'more productive' working in JavaScript or Java. They would determine this based on the ratio of lines of code written to the time taken. They then drew conclusions from their findings. However, there were several issues in this experiment. I have outlined a few of them below.

Issues

- ***Construct validity***

1. **"More Productive"**
   The title could be interpreted as meaning the type of person that uses is behaviourally less likely to be inefficient. The choice of the words "more productive" is unclear and misleading. The researchers should aim for a higher standard of clarity. "Developers are more efficient at writing code in JavaScript than Java" would be a clearer title. But even this has issues (see the next point).

2. **"Developers" versus "JavaScript students"**
   The study's conclusions are about 'developers', but the 5 subjects were all students. With a large sample size, one could reasonably validate the use of students, but the paper has neither a large sample size nor a validation. The best fix would be to use a larger sample size of developers, or a larger sample size of students with a good validation as to why it is still representative.

- ***Internal validity issues***

3. **The measurement of time**
   This measurement is inclusive of thinking time, question time, and procrastination. These will differ between participants and languages and hasn't been adequately addressed. The solution is to only include the duration spent directly working on the code because it more accurately represents the general time it takes to complete code.

4. **Assuming data**
   Researchers assumed a time for incomplete tasks. And while they did not specify (reliability issue), they must also assume the lines of code the student would have written. This introduces an unacceptable amount of bias as the researcher assumes the task would be completed as they themselves would have completed it. A simple solution would be to have a larger sample size and only include participants that complete the task while indicating the completion rate.

5. **Attempt per user**
   It is not stated how many times the users are tested (reliability issue), but the assumption is students had one attempt for each task. As participants are only students, it may have been better to do a few similar tasks and take the average coding efficiency per participant, or their best attempt, to better approximate a true average coding efficiency for the cohort.

6. **Variable skill levels**
   The variation in skill level makes individual results incomparable, meaning the calculated average may not accurately describe the cohort or any subgroup. Students with little experience will not approximate experienced developers well (external validation issue). To solve this, separate the cohort based on grade or some other metric, calculate cohort averages, and then validate how skilled students are representative of developers in the field.

7. **Very small sample size**
   It is difficult to determine statistical significance with such a small sample size as the central limit theorem mightn't hold true. A larger sample size is needed in order to confidently make conclusions that extrapolate well. The solution is to have more participants in the experiment.

8. **Variance in observer assistance**
   The treatment of each student cannot be identical, adding unaccounted for variables. The simplest solution is to only include the duration of time spent directly working on the code. This removes that variability and ensures the participant properly understands the advice before continuing with the code which better represents a developer's efficiency.

9. **Results are 2/5ths assumed**
   Two of five participants didn't finish the Java task, meaning 40% of the data used to calculate the average is assumed. The researchers have heavily influenced the findings of the experiment by including these fictitious data points. It would be better to exclude them and comment on the number of students that were unable to complete the task.

10. **Variable selection/ identification issues**
    The researchers have not taken into consideration variables that may influence the efficiency of a language, such as hardware, optimal performance times, etc. They have not adequately isolated the variables of interest, accounted for other variables, or commented on these factors. This issue can be avoided if the researchers are more specific about their research question. A more research question would require more specific evidence.

- External validity issues

11. **Inherent advantage to JavaScript efficiency**
    Participants are more proficient in JavaScript than Java. The impact this would have on the efficiency in JavaScript has not been accounted/commented on. Additionally, comparing an individual's results does not extrapolate on to the wider population of programmers well. Instead, researchers should have taken students with no prior skills in either language or compared efficiencies of JavaScript programmers with a similarly experienced population of Java users.

12. **Incomparable conditions (internet access, syntax assistance, etc.)**
    Assessing coding efficiency without all the tools that developers use may result in untrue or unjustified conclusions. If there is more support online, or more characters written with syntax assist, in one language over the other, then the time taken to complete a task will be reduced. The researchers should have allowed these tools, as it better represents the real-world setting.

13. **White space inclusion will create misleading results**
    If the users aren't told about this, then individual style is being considered and advantaging those that put thought into clean coding. Additionally, does this mean that blank lines are treated as lines of code? If so, then 'neat coding' is being rewarded unnecessarily. Blank spaces used for stylistic purposes should be excluded from measurement. A simple solution would be to exclude 'unnecessary' white spaces. Excessive use of white spaces should be penalised as it would be expected to reduce efficiency.

14. **Individual efficiency is not team efficiency**
    There are aspects of development teams that may make a seemingly less efficient language functionally more efficient. For example, a suboptimal language may be more efficient in a practical sense because it's easier for other developers to interpret and directly results in projects being completed faster. The researchers should have only commented on individuals.

- Reliability

15. **Link may become outdated**
    Regarding the hyperlink in the code, this link or address may change over time. The researchers should provide a method to view the page as they themselves viewed it at the time of writing.