1. Bayes’ Theorem states that the probability of an event A (in this case, that the programmer wants to write some particular identifier) given an event B (in this case, the code that they have previously written) is proportional to the probability of event B given event A times the probability of event A. For example, consider a class *MyClass* with two static methods defined on it – *run*, and *walk* – and the user has typed:

*MyClass.*

The editor might approximate the two relevant probabilities for each of the possible identifiers. Since P(*MyClass.* | *run*) = P(*MyClass.* | *walk*) so the two suggestions should be ordered based on which method is more commonly used. If, however, the programmer types:

*MyClass.r*

Then P(*MyClass.r* | *run*) > P(*MyClass.r* | *walk*) so the *run* identifier should be suggested first, unless P(*walk*) is so much greater than P(*run*) that it is likely that a typo occurred.

1. One way to empirically measure the improved efficiency is to observe programmers writing code with and without the software. You could measure the number of lines written in a given amount of time in both cases (likely a Poisson distribution).

However, this would require a very large sample, as the data would be very noisy. This is due to the fact that programming efficiency with respect to lines-per-hour can vary significantly depending on the nature of the program. For example, if one program is very complex and requires a lot of careful thought, and another is largely boilerplate (both of which are extremely common in Java) then the difference in efficiency likely only be due in small part to the editor itself.

One way to combat this effect is to only consider time during which the programmers are actively typing, though this would need to be rigorously defined and carefully measured.

Another way would be to ask participants in the study to all write the same set of programs both with and without the software.

1. Small variance and small effect size likely means that the software has little impact on efficiency for anybody.

Small variance and large effect size could indicate that the software broadly improves everybody’s efficiency. This means that the software might be helpful to include by default in IDEs.

Large variance and small effect size suggests that overall, the software didn’t affect most people’s efficiency, but massively helped a few people. This could suggest that the software needs to be easier to learn or more intuitive, as the data suggests there is a large efficiency increase available to those who know how to take advantage of it. It might instead suggest that the software is helpful only in a few circumstances (e.g., its predictions are very good for working with OpenGL libraries but not so much for standard Java programming). In this case it might be useful to try to make the software more generically applicable.

Large variance and large effect size means that overall, the software helps most programmers, but some find it to make little difference. This might indicate that the software is not suited for certain tasks but is overall fairly useful. The developers should do further research into what causes the low-effect results.

1. One way in which a programmer would decide whether or not to use this feature is how easy it is to integrate this feature into their workflow. For example, if the functionality can simply be enabled and disabled at any time, then the user will be less likely to fear viscosity and premature commitment – they would be able to change their mind if they don’t like it. This could be made clear to the programmer with a simple label such as “This can be disabled at any time”.

Programmers’ adoption of IDE features is also largely subject to trend or word-of-mouth. If they see trusted friends or internet sources recommending the feature, they will be more likely to do the same. Furthermore, if the feature is hidden away in some settings menu, fewer people are likely to find and use it. However, if users are given a clear Yes/No choice about whether to enable the feature (for example, when creating a new project), some will choose Yes, and if they like it, others will follow the trend.