Ludwig-Maximlians-Universität München
Department für Psychologie

Lehr- und Forschungseinheit Psychologische Methodenlehre und Diagnostik

Implementation of a Formal Model for Scientific Errors in Data Analysis

Bachelorarbeit zur Erlangung des akademischen Grades
Bachelor of Science im Fach Psychologie

Eingereicht von Keno Mersmann
Matrikelnummer 12413065

Betreuer: Prof. Dr. Felix Schönbrodt

München, den 20.08.2024

Implementation of a Formal Model for Scientific Errors in Data Analysis

Mersmann, Keno [1]

[1] Department of Psychology

Ludwig-Maximilians-Universität München

Abstract

Scientific errors can occur at various stages of the research process, with data analysis in empirical research being particularly susceptible to such errors. Unintentional human mistakes during this stage can lead to incorrect numerical results, which may subsequently distort data interpretation. This thesis reviews a formal model proposed by Kohrt et al. (2023) and further specified by F. Melinscak, which aims to explain the emergence of faults in data analysis, the process by which researchers identify and attempt to correct these faults, and the implications for published literature where such faults remain undetected up to and post publication.

The core contribution of this thesis is the implementation of this model in **`R`** through the **`scerrModel`** package (https://github.com/advieser/scerrModel). It enables the simulation of entire literature datasets based on the model, facilitating the analysis model behaviour as well as the distribution of error sizes and the number of faults that persist in studies after undergoing a correction process. A preliminary analysis of the model's behaviour is conducted using this implementation, offering initial insights into its dynamics. The thesis concludes with a discussion of potential directions for further development and refinement of the model.

Implementation of a Formal Model for Scientific Errors in Data Analysis

Contents

Introduction **7**

Literature Search for Other Formal Models of Scientific Errors 8

Review of a Formal Model of Scientific Errors **9**

Terminology 10

Formal Model 10

Modelling Scope and Goal 10

Structure and Assumptions of the Model 11

Mathematical Specification 12

Data Generating Process of Incorrect Results 13

Agents' Decisional Model for Performing Another Search 15

Agents' Mental Belief Updating Process 20

Summary 25

Presenting `scerrModel` – an implementation in R **25**

Functionalities and Standard Workflow 26

Options for Seed Setting 29

Example Simulation of One Literature Dataset 30

Analysis of Model Behaviour and Sensitivity Analysis **39**

Determinants of Decision to Stop 39

Benefit, Cost and Resources 40

Subjective Probability of Finding a Fault in the First Round 40

Relation between Number of Code Units, Probability of Faults, and Error Sizes 45

Pattern in Belief about the Number of Remaining Faults 46

Discussion 48

Open Questions Concerning Model Assumptions 49

Defining Code Units 49

Distributional Assumptions 50

Directions and Difficulties for Empirical Verification 51

Improvements for the Implementation 51

Conclusion 52

References 54

Introduction

"To err is human; to persist in error is diabolical." This quotation, attributed to the Roman philosopher Seneca, underscores the inevitability of mistakes in human endeavours, including science, where the consequences of unrecognized or uncorrected errors can significantly compromise the validity and reliability of findings. Such scientific errors, diverse in nature, lead some authors to conclude that “most published research findings are false” (Ioannidis, 2005) and empirical reports on the low replicability of studies (e.g. Open Science Collaboration, 2015) do little to alleviate these concerns.

The literature presents various classifications of scientific errors. For example, van Ravenzwaaij et al. (2023) distinguish between two types: the first involves errors that impact scientific outcomes due to non-content-related factors, while the second category encompasses errors arising from inaccuracies and mistakes within the research process itself. When analysing errors that led to retraction, Andersen and Wray (2019) put forth a taxonomy of errors that differentiates between three levels: (1) errors occur at the level of the material, (2) errors in the data, and (3) errors occur at the level of the analysis or interpretation of the data.

Both classifications encompass errors that occur during data analysis, a critical subset of scientific errors. Given that data analysis is a fundamental component of all empirical research, errors in this stage are particularly consequential. Such mistakes can lead to incorrect numerical results, which may not only distort the interpretation of findings but, in some cases, entirely alter the conclusions drawn from the data.

However, there is limited research on the numerical impact of such errors. To address this gap, Kohrt et al. (2023) proposed a theoretical model aimed at understanding the emergence and potential correction of analysis errors in empirical research. The model introduces a set of assumptions about the data-generating process that leads to incorrect results and outlines a potential mechanism for detecting and rectifying errors that occur during data analysis. F. Melinščak (personal communication, 2023) has developed a

specification of this theoretical model that formalises its concepts within the framework of probability theory.

Formalisation offers several significant advantages. By requiring the explication of all assumptions underlying a model (or, more broadly, a scientific theory), it promotes a transparent and critical examination of the proposed mechanisms. Additionally, formalization allows for a more rigorous assessment of the model's validity (Borsboom et al., 2021; van Dongen et al., 2024).

In this thesis, I will review the model and its specification, and present an implementation in **R** that simulates the process described by the model, generating a dataset representing literature that has undergone this process. This implementation enables the analysis of the distribution of error sizes and the number of faults remaining in a data analysis after correction, up to the point of publication.

This thesis is organized as follows: Initially, a comprehensive review and detailed explanation of the underlying formal model will be provided. Subsequently, the **scerrModel** package, which serves as an implementation of this model, will be introduced. The structure and core functionalities of the package will be outlined, followed by the presentation of a simulation example to elucidate the model's application and capibilities. Following this, an analysis of general tendencies within the model's behaviour and short sensitivity analysis will be conducted. Finally, the thesis will conclude with a discussion of the results from these analyses, including an assessment of the model's current state and potential future improvements. Directions for ongoing work, as well as possible avenues for empirical verification and calibration of the model, will also be highlighted.

Literature Search for Other Formal Models of Scientific Errors

Before presenting the formal model of analysis errors that is the foundation of this thesis, I would like to give an overview over other formal models of scientific errors. However,

a literature search (that is detailed below) yielded no findings in this area, highlighting the need for filling this gap in the scientific discourse.

A literature search was performed using Web of Science with the search string TS=(("formal model" OR "data generating process" OR "dgp") AND ("scientific error" OR "scientific errors" OR ("error" AND "science"))), returning 11 results. Additionally, backward reference searching was attempted in known articles about scientific errors, with no success.

Review of a Formal Model of Scientific Errors

This section introduces the formal model that serves as the foundation for the implementation. Its aim is to model the emergence and correction of analysis errors in empirical research.

This section begins by delineating the used terminology throughout the rest of this thesis. The remainder of this section can be split into two parts: The first part summarises the original model presented by Kohrt et al. (2023), providing a comprehensive overview that starts with defining the model's scope, followed by an outline of its core assumptions. Based on these assumptions, the model is divided into two primary components: the data-generating process of incorrect results and the agent's decision-making model for identifying faults. This section also introduces a mental model that updates the agent's beliefs regarding faults and effect sizes. Part of this second part is a mental model updating the agents beliefs concerning faults and effect sizes. The second part of this section details the mathematical specification of the model with its additional assumptions, made by F. Melinščak (personal communication, 2023), which follows the same differentiation into two components. This specification will serve as the basis for the subsequent implementation in R.

Terminology

I will adhere to the terminology established by Kohrt et al. (2023). Specifically, in the context of incorrect results, the following distinctions will be made: “a human action (a mistake), its manifestation (a hardware or software fault), the outcome of the fault (a failure), and the magnitude of the incorrect result (the error)” (International Organization for Standardization, n.d., p. 275). Additionally, the simplified representation of a researcher, as posited in the formal model, will be referred to as an “agent.”

Formal Model

Modelling Scope and Goal

While mistakes can happen in different stages of the scientific process (see Introduction), the formal model described here limits itself to the stage of data analysis in empirical research. In this context, data analysis is understood as a narrower concept than some might understand it to be. While one could, for example, consider the choice of statistical method, e.g. using a linear regression, and possible mistakes related to said choice, to be part of the analysis, here it is understood solely as the process of performing the analysis itself, using a software designed for this purpose, such as the **R** programming language, that performs computations to return a numeric result for the specified analysis.

Furthermore, the aim is to consider only those mistakes, that numerically impact the result of an analysis—specifically, mistakes that change the numeric value of the analysis result. This focus is justified since in many statistical analyses the interpretation is based on the precise realisation of specific values, such as p-values, t statistics, or effect sizes. Therefore, mistakes that influence the numerical result may inadvertently influence its interpretation which is a central objective of empirical research.

These mistakes might, in the simplest of cases, involve unintentionally placed brackets or incorrect binary operators. However, these should only be considered as examples, and many other types of mistakes are possible. It is important to note, however,

that mistakes which would prevent the analysis from being completed—such as those that would trigger an error—are assumed to always be corrected before publication.

In summary, the scope of the present model is to model the emergence of mistakes in data analysis for empirical research that have a numerical impact on a single computational result. By focusing on this subset of potential mistakes, the model seeks to describe how errors in the computation of a numeric result may stay undetected up to and post publication and may thus propagate to the result that is used for interpretation in a published scientific work.

Structure and Assumptions of the Model

The model can be divided into two parts. First, the data generating process (DGP) of incorrect results and second, the process of detection and correction of incorrect results that an agent goes through during data-analysis. I will now describe both components and summarise the assumptions made for each. Through these assumptions the model is defined.

For the data generating process of incorrect results, it is assumed that an agent initially makes a finite number of mistakes, each of which contributes to the initial error of the computational result, i.e. the numeric value by which the result is incorrect. For the process of detection and correction, it is assumed that agents perform a search for faults in rounds. If they find a fault, they will correct it, removing its contribution to the total error of the computational result. During this process they will not introduce any new faults. Agents have access to an initial amount of resources, from which in each round a cost is subtracted. However, finding a fault will also entail a benefit, if it happens. These resources might for example be time or money, however, the definition is purposely left abstract.

Each round, the agent must decide whether to continue searching. This is dependent on three separate mechanisms: First, the remaining resources must cover the cost of another search round. Second, the potential costs and benefits of performing another search round will be weighed against each other. Third, the agent’s belief about finding another fault

in the next search round must be high enough. This belief is influenced by the plausibility of the current numerical result, from which all previously found errors have been removed, which itself depends on the agent's prior expectations about plausible values for the numerical result as well as the belief about the data generating process of actual faults. This belief about the DGP is assumed to be an approximation of the real DGP.

Mathematical Specification

The mathematical specification made by F. Melinščak (personal communication, 2023) takes the assumptions listed above and combines them through the definition of different random variables into a stochastic model that follows the same differentiation into two parts.

In this section, different probability distributions will be defined where I will adhere to the following notation: The index of parameters will indicate their corresponding random variable, and the superscript of parameters is a label to clearly assign it to either the objective reality (superscript (o)) or the subjective beliefs of an agent (superscript (s)). The superscript above a tilde may be either obj or $subj$ also indicating whether we are talking about the true distribution in reality or the agent's belief about that distribution. All defined random variables are summarised in Table 2 which can be found at the end of this section.

The first additional assumption is that the computational numeric result of the analysis is understood to be an effect size. This is a sensible choice as effect sizes are very informative for interpretation and their reporting is at least encouraged if not required in several different scientific fields, e.g. in biology (Nakagawa & Cuthill, 2007), and not least of all in psychology (American Psychological Association, 2020; Lakens, 2013). For further simplification, one might think of Cohen's delta as an example when reading the following sections.

Data Generating Process of Incorrect Results

A single result can be considered correct if its calculation matches the intention of the researcher performing the analysis. For the present specification, this correct result is the true effect size of a study (or a part of a study, in cases where one study would report multiple effect sizes, as is common practice). This true effect size is defined as a random variable, called δ, that is drawn from a parametric probability distribution. This implies nothing else but that different true effect sizes δ may be differently probable. How probable which true effect size is depends on the type of probability distribution and its parameters.

For the true effect size, we assume a normal distribution, indicating that effect sizes may be positive or negative and smaller effect sizes are considered more likely than larger ones. At this point, this choice is largely made arbitrarily and not based on theoretical or empirical considerations. Formally, we write

$$\delta \overset{obj}{\sim} Normal\left(\mu_{\delta}^{(o)}, \sigma_{\delta}^{2\,(o)}\right)$$

where $\mu_{\delta}^{(o)}$ is the expected value and $\sigma_{\delta}^{2\,(o)}$ is the variance of the distribution.

Having defined the true effect size δ of a study, we now want to define possibly erroneous effect sizes that represent the current computational result for each round the agent performs. This result will contain all undiscovered errors which will exist because of the initial finite number of mistakes the agent is assumed to make.

To put this into mathematical terms, we make an additional assumption, namely that the analysis plan can be represented as a list of N discrete steps. We call these steps *code units* as this seems most intuitive in the context where the analysis is performed using a programming language, i.e. where the analysis can be represented through code. As of right now, these code units remain an abstract concept that lacks a definite analogue in reality. However, one might want to conceptualize these as lines of codes or analytic transformations in the code.

It is then reasonable to assume that each of these N code units contribute to the result of the computation and may possibly contain a fault. Formally, we define binary fault

indicators e_i that indicate whether the i-th code unit contains a fault, where i can indicate any of the N code units, i.e. $i = 1, \dots, N$. We presume that these fault indicators are generated through a Bernoulli distribution with $\pi_e^{(o)}$ being the probability of a fault accruing in a code unit. We write

$$e_i \overset{obj}{\sim} Bernoulli\left(\pi_e^{(o)}\right).$$

In addition, we define u_i as error sizes that give the possible error associated with the i-th code unit. We again assume that these follow a parametric probability distribution, which in this case is presumed to be a normal distribution. We write

$$u_i \overset{obj}{\sim} Normal\left(\mu_u^{(o)}, \sigma_u^{2(o)}\right).$$

It should be noted that this is a strong assumption to make as there is no theoretical or empirical a priori reason to assume that the numerical impact of software faults would follow this specific distribution. Furthermore, note that we write the parameters $\pi_e^{(o)}$, $\mu_u^{(o)}$ and $\sigma_u^{2(o)}$ without a subscript i in the index to indicate that all fault indicators and error sizes come from the same probability distribution, respectively. This means that the parameters are same for all code units.

As explained in the previous section, the agent will be performing a search for faults in rounds. The specification supposes that the agent will go through one code unit per round looking whether a fault is present in that code unit or not. This implies that the agent will only be able to perform a maximum of N search rounds, after which they will have checked the whole source code for faults. Because of this, we can take i to also indicate the current search round, meaning that e_i and u_i are the fault indicator and associated error size of the code unit examined in the i-th search round.

Putting all these definitions together, we can now define the effect size computed before the next search round, i.e. with all faults removed that were found in previous rounds. We denote the current computed effect size before the i-th search round as d_i which can be calculated as

$$d_i = \delta + \sum_{j=i}^{N} e_j u_j \, .$$

As an example, we can look at the computed effect size before any rounds searching for faults have been performed. As this is the effect size before the first round we set $i = 1$ and can then see that d_1 is calculated through

$$d_1 = \delta + \sum_{j=1}^{N} e_j u_j \, .$$

Note that we have defined e_i to be binary, either indicating that a fault is present in the i-th code unit or not. While the error sizes are almost always non-zero, through building the product of the two, a code unit only introduces a distortion to the true effect size if a fault is present in that code unit (i.e. if $e_i = 1$).

In summary, this means that the initial computed effect size is the true effect size plus the sum of all error sizes u_i of faulty code units indicated by e_i. Going through the search rounds, the summation index j will increase, meaning that only the faults in the i-th code unit and all later code units will influence the computed result before the i-th round is performed.

Agents' Decisional Model for Performing Another Search

As stated in the previous section, the agent will have to decide each round whether they want to continue (or in case of the first round, begin) searching for faults or stop (or in case of the first round, not begin) the search. Termination of the search is taken to mean that all undiscovered faults remaining in the code will stay undetected and persist up to and post publication.

Before making this decision, the agent will know what resources they have remaining. They will know what costs another search round would entail, and what benefit another fault would bring. Moreover, we assume that stopping the search will neither entail any cost nor bring any benefit. They will also have observed the current effect size, which in the first

round is the initially computed effect size that includes all errors of faulty code units and in later rounds will be this initial effect size minus all potential errors of code units that have already been checked.

The decision to continue or stop searching will then depend on (1) whether enough resources remain for another cost entailing search round and (2) on the costs and benefits of another search round weighed by the plausibility of the current observed effect size.

While the first part of the decision-making process needs no further explanation, the second part needs to be explicated. Note also, that we here combine two mechanisms, that were postulated separately in the original model, into one that will now be described in more detail.

To model the weighing of options in the second part of the decision-making process, the specification employs the principle of choosing the action that maximises the expected utility for the agent, formalised by Savage (1954) and used in Bayesian Decision Theory. In this theoretical framework probabilities are understood as numerical expressions of subjective beliefs about a decision-maker's choices. For a historical overview and mathematical introduction, see Karni (2008), for another introduction more accessible to the interested reader without a strong mathematical background, see Lindley (2000). The basics of this principle, as needed for the understanding of the present model, will be explained in the following paragraphs.

The agent can decide between two actions: continuing to search for faults or stopping the search. For now, these two options will simply be referred to as "search" and "stop". Each of these actions has the same two possible consequences, either finding a fault or not finding a fault, which for now we simple refer to as "fault" and "no fault". Formally, an action is defined as a random variable with the set of consequences being its support, i.e. the possible realisations of that random variable.

A utility function, denoted $U(\cdot)$, then ascribes a utility value to each consequence. In the present case, not finding a fault has the utility of (negative) costs, since a search was

performed incurring that cost while no fault was found resulting in no benefit. However, finding a fault has the utility of bringing a (positive) benefit, while still entailing a (negative) cost, since a search round was performed.

The subjective expected utility is then defined as the expected value of the utility of the action, where the probabilities used to calculate the expected value are the subjective belief of the agent that the action will result in the respective consequence. Formally, we write the subjective expected utility for an action A with consequences a_1 and a_2 as

$$E\big(U(A)\big) = \sum_{i=1}^{2} U(a_i) \cdot P\big(U(a_i)\big)\,.$$

The formula used here is simply the formula for the calculation of the expected value of a discrete random variable, that has been transformed by the function U.

Transferring this to the present use case, we can write the subjective expected utility for both possible actions the agent might take, search and stop, as

$$E\big(U(search)\big) = (benefit - cost) \cdot P(fault) - cost \cdot P(no\ fault)$$

$$E\big(U(stop)\big) = 0 \cdot P(fault) + 0 \cdot P(no\ fault) = 0\,.$$

The first equation can be simplified to

$$E\big(U(search)\big) = P(fault) \cdot benefit - cost\,.$$

Following the principle, the agent will choose the option that maximises the subjective expected utility. In the case here, where only two options exist, this means, that the agent will choose the options that has the higher expected utility, i.e. they will choose to continue searching if

$$E\big(U(search)\big) > E\big(U(stop)\big)$$

which is equivalent to saying

$$E\big(U(search)\big) > 0\,.$$

This finally leads to the decision rule

$$P(fault) \cdot benefit - cost > 0$$

which can be reformulated as

$$P(fault) \cdot \frac{benefit}{cost} > 1 \, .$$

The benefit-cost-ratio appearing in the above equation is understood as a fixed parameter of the analysis.

The above formula now gives a concrete decision rule to determine in which cases an agent would decide to continue searching for faulty code units, given that enough resources are remaining. However, as the agent will have to make that decision each round, we are now faced with the challenge of calculating the probability of finding a fault in the next round, for each round. To formalise this, some additional constructing steps are necessary.

First off, we must formalise what we have been referring to as $P(fault)$, the probability of finding a fault in the next search round. We have already noted that before making the decision, the agent has observed the current computed effect size d_i. Furthermore, what we are ultimately interested in is whether the next code unit contains a fault, which would be indicated by the fault indicator for the i-th code unit realising as 1. Since we assume that the agent will also depend his choice on the plausibility of the current observed effect size, it would only make sense to condition the subjective belief of finding another fault in the next search round on the current observed effect size. However, it would also be logical for the agent to consider the number of faults they have already identified, i.e., the realisation of all previously observed fault indicators.

Thus, we formalise the probability of finding a fault in the next (i-th) round as $P(e_i = 1|O_i)$, where O_i is the set of all observations made prior to the i-th search round, which we formally write as $O_i = d_1, \dots, d_i, e_1, \dots e_{i-1}$. Note that we include all previously observed effect sizes purely for convenience, so that the set of previous observations can be built up iteratively over each round. Furthermore, be aware that the index i is no longer consistent in its meaning for all the different random variables. The current effect size d_i is the effect size *before* the i-th round, the set of observations O_i is the set of all effect sizes and fault indicators *before* the i-th round, while the fault indicator e_i is the indicator for the code block *in* the i-th round. This is why the set of observations only includes the fault

indicators up to e_{i-1}. Lastly, also note that the index i is defined as $i = 1, \dots, N$, and in the case of the first round (with $i = 1$), where no previous search round was performed, e_{1-1} does not exist, per definition of the index and is thus not part of the set of previous observations In the first round O_i would only include the intial computed effect size d_1.

Having now formalised this probability, we are still faced with the difficulty of computing it for each round. At this point, another assumption has to be made. For this we are again effectively only shifting the problem to another step in the modelling process. However, this will be final such shift.

We are defining a new probability distribution for the subjective belief about the number of remaining faults c_i before the i-th search round. We use this new distribution to define the probability of finding a fault in the next round as

$$P(e_i = 1|O_i) = \frac{E(c_i|O_i)}{N - (i - 1)}.$$

Here, the numerator is the the conditional expected value of remaining faults c_i given the set of observations O_i. As the distribution over c_i given O_i represents the subjective belief of the agent, the expected value can be interpreted as the number of remaining faults the agents expects in the yet unchecked code units (which, as a reminder, includes the i-th code unit). The denominator gives the number of code units that haven't been checked for faults yet. This quotient can be interpreted to mean that the agent always assumes that, given an expectation of how many faults remain, it is equally likely to find a fault in any of the remaining code units.

The next section describes, how this belief distribution for the number of remaining faults will be updated each round, given the prior expectations of the agent and the plausibility of the current observed effect size, through a generative process using Bayesian updating.

Agents' Mental Belief Updating Process

In this section, I will summarise the definition of the generative process for the subjective belief of the agent for the number of remaining faults before the i-th search round, denoted as c_i. This process will use Bayesian updating for which we need to make additional assumptions about the prior beliefs an agent holds before starting the analysis. I will first explain the updating of the agent's belief conceptually, before giving the mathematical definition, which may be skipped on first read-through.

Conceptual Explanation. The agent holds three different kinds of prior believes which will each be modelled as (subjective) probability distributions. First, the agent holds an initial belief about the occurrence of a fault within a code unit, i.e. about the fault indicators e_i. We model this in such a way that the agent also holds a belief about the probability of a fault occurring, allowing us to model the agent's uncertainty about said probability. From this we can derive an initial belief about the number of remaining faults which will then be updated through consecutive search rounds. Second, the agent holds a belief about the true effect size. They may for example expect an effect size of 0.5 with some degree of uncertainty. Third, the agent holds a belief about the size of errors associated with faulty code units. We will construct these last two distributional assumptions in such a way that we can use them to define the plausibility the agent would assign an observed effect size d_i, given the belief about the number of remaining faults. It should be highlighted that we suppose that the agent's belief about the true effect size as well as the effect sizes will not be updated during the whole process. In this sense, the agent's mental model must be considered irrational.

Finally, with these distributional assumptions it is possible to set up a generative process to update the agent's belief about the number of remaining faults given observations made up to that point in time. For this we use Bayesian updating, a modelling approach where Bayes' rule is used to update the belief about the number of remaining faults given the plausibility (or likelihood) of an observation occurring, weighed by the prior belief about

the number of remaining faults. We will use this mechanism twice per round, once where the observation is the current computed effect size and once where the observation is the fault indicator of the code unit examined in that search round. With this, we update the belief about the number of remaining faults which will then be used to calculate the belief about finding a fault in the current round, as described in the last section.

Mathematical Construction. Let's now go over the mathematical construction of this process. First, we assume a Bernoulli distribution with the parameter $\pi_e^{(s)}$ for the belief about a fault occurring in the i-th code unit. We write

$$e_i \overset{subj}{\sim} Bernoulli\left(\pi_e^{(s)}\right).$$

However, the agent may also have some uncertainty about the probability of a fault occurring in a code unit, given by $\pi_e^{(s)}$, which is why we model this belief as a separate probability distribution, namely a Beta distribution which has two shape parameters, $\alpha^{(s)}$ and $\beta^{(s)}$, and has support over the interval from zero to one, making it suitable to model probabilities. Formally, we write

$$\pi_e^{(s)} \overset{subj}{\sim} Beta\left(\alpha^{(s)}, \beta^{(s)}\right).$$

These two distributions can be used to model the initial belief about the number of remaining faults c_i before the i-th search round. Since the agents expects the fault indicators to follow a Bernoulli distribution, it follows that they would assume that the sum of fault indicators, i.e. the number of remaining faults, would follow a Binomial distribution. Combining this with the belief about the probability of finding a fault in a single round that is expressed through the Beta distribution, we get a Beta-Binomial distribution for the belief about the number of remaining faults c_i, i.e.

$$c_i \overset{subj}{\sim} BetaBinomial\left(N-(i-1), \alpha^{(s)}, \beta^{(s)}\right).$$

Second, we assume for the belief about the true effect size that it would follow a normal distribution mirroring the true distribution of the true effect size as it is assumed to

approximate the process in the objective reality, however using other (subjective) parameters. We write

$$\delta \overset{subj}{\sim} N\left(\mu_\delta^{(s)}, \sigma_\delta^{2(s)}\right).$$

Finally, the agent holds a belief about the distribution of error sizes u_i, for which we assume the same distributional family as in the objective reality, i.e. a normal distribution, for the same reasons as stated above. We write

$$u_i \overset{subj}{\sim} N\left(\mu_u^{(s)}, \sigma_u^{2(s)}\right).$$

Now, we can model the belief about the current observed effect size d_i, given the belief about the number of remaining faults c_i, as the sum of (a) the belief about the true effect size, and (b) the belief about the error sizes times the number of remaining faults as

$$d_i|c_i \overset{subj}{\sim} N\left(\mu_\delta^{(s)} + c_i u_u^{(s)}, \sigma_\delta^{2(s)} + c_i \sigma_u^{2(s)}\right).$$

This holds since we assume a normal distribution for δ as well as u_i, and since the sum of two normally distributed random variables will still be normally distributed. This property of the normal distribution holds for all so-called stable distributions. For this reason, other stable distributions, such as a Cauchy, could be used here instead. To summarise, we have now modelled the beliefs about the number of remaining faults c_i and about the current observed effect size d_i, given the belief about the number of remaining faults c_i.

Finally, we can combine these results to define the belief updating process. For this, we will first go other the initialisation of the process in the first round before describing the process in an arbitrary (i-th) round.

At the beginning of the first round, the agent observes the initially computed effect size d_1. They will now have to make the decision whether to search for a fault or not, based on their belief of finding a fault in the first search round ($P(e_1 = 1|O_1)$, where $O_1 = d_1$). This belief is calculated using their belief about the number of remaining faults c_1 given O_1. So, they will have to update that belief given the initial effect size they observed, d_1. This can be

formalised using Bayes' rule. The prior, the belief about the number of remaining faults c_1, is expressed through

$$c_1 \sim BetaBinomial\left(N, \alpha^{(s)}, \beta^{(s)}\right),$$

and the likelihood of observing that initial effect size d_1 given the agents belief about the number of remaining faults c_1 is expressed through

$$d_i|c_i \sim N\left(\mu_\delta^{(s)} + c_i u_u^{(s)}, \sigma_\delta^{2(s)} + c_i \sigma_u^{2(s)}\right).$$

This results in the following equation:

$$P(c_1|d_1) \propto P(d_1|c_1) \cdot P(c_1)$$

where $\propto$ stands for proportionality which is generally sufficient for expressing Bayes' rule. Since $P(c_1|d_1) = P(c_1|O_1)$, the resulting distribution can be used to compute the expected value $E(c_1|O_1)$, which is then used to calculate $P(e_1 = 1|O_1)$.

Should this probability be high enough for the agent to decide to search for a fault, they will observe the fault indicator e_1 of the first code unit. Based on its realisation, the agent will update their belief about the number of remaining faults for the next round c_2, which will be used as a prior in the following round. For this, two steps are necessary. First, the belief over c_1, given both d_1 and e_1, will be updated using Bayes' rule again, before secondly, the belief for c_2 will be calculated based on this updated belief.

The second application of Bayes' rule is as follows:

$$P(c_1|d_1, e_1) \propto P(e_1|c_1, d_1) \cdot P(c_1|d_1)$$

Here, the prior $P(c_1|d_1)$ was calculated in the first application of Bayes' rule while the likelihood $P(e_1|c_1, d_1)$ is assumed to come from

$$e_1 \sim Bernoulli\left(\frac{c_1}{N}\right).$$

This is assumed to be case because of conditional independence, i.e. $P(e_1|c_1, d_1) = P(e_1|c_1)$. However, this has not been verified in the current specification of the model.

Finally, the belief about the number of remaining faults in the next round c_2 is calculated via the recursive relationship $c_2 = c_1 - e_1$, where $c_2 = c_1$ if $e_1 = 0$, which can be

interpreted to mean that the belief about the number of remaining faults will not change in this step, if no fault was discovered in the first code unit. Formally, this is expressed through

$$P(c_2 = x|d_1, e_1) = P(c_1 - e_1 = x|d_1, e_1) = P(c_1 = x + e_1|d_1, e_1)\,.$$

which means that $P(c_1|d_1, e_1)$ gets shifted one to the left, if $e_1 = 1$. Here, c_2 has support on $[0, N-1]$.

For completeness' sake, this will now be transferred to the case of the i-th round. At the beginning of the i-th round, the agent observes a calculated effect size d_i. Now, the agent updates their belief about the number of remaining faults c_i given the all prior observations O_i, which by definition includes the newly observed effect size d_i. Remember that O_i is defined as $d_1, \dots, d_i, e_1, \dots, e_{i-1}$.

This updating can be formalised using Bayes rule

$$P(c_i|d_i, O_{i-1}, e_{i-1}) \propto P(d_i|c_i, O_{i-1}, e_{i-1}) \cdot P(c_i|O_{i-1}, e_{i-1})$$

where formally $O_i = d_i, O_{i-1}, e_{i-1}$. Here, $P(d_i|c_i, O_{i-1}, e_{i-1}) = P(d_i|c_i)$ since it is assumed that the beliefs about δ and u_i are not being updated throughout this process. $P(c_i|O_{i-1}, e_{i-1})$ is the belief about the number of remaining faults updated at the end of the last round.

The agent now makes his decision whether to continue searching or not. Should they do so, the belief about the number of remaining faults is updated according to the observed fault indicator e_i, i.e.

$$P(c_i|O_i, e_i) \propto P(e_i|c_i, O_i) \cdot P(c_i|O_i)$$

Here, $P(c_i|O_i)$ was calculated in the first application of Bayes' rule and $P(e_i|c_i, O_i)$ is assumed to follow a Bernoulli distribution, specified by

$$e_i \sim Bernoulli\left(\frac{c_i}{N - (i-1)}\right),$$

because of conditional independence, i.e. $P(e_i|c_i, O_i) = P(e_i|c_i)$. This is the generalisation of the formula introduced for the first round that has not yet been verified.

Finally, the prior for the next rounds, the belief over c_{i+1}, is updated, i.e.

$$P(c_{i+1} = x|e_i, O_i) = P(c_i = x + e_i|e_i|O_i)\,.$$

where c_{i+1} has support on $[0, N-i]$. This finalises the definition of the updating process, meaning that the whole specification is now described.

Summary

In this section, the scope and assumptions of the formal model have been explained before its mathematical specification was detailed. In short, an agent makes a number of mistakes during data analysis that has the goal of returning a single effect size. Through these mistakes, faults are introduced to the code. Based on the returned effect size and the amount of remaining resources an agent can decide whether to search for faults or not, which they will do in rounds. They stop their search for faults if a set of conditions is no longer met. All faults that might remain in the code will stay undetected up to and post publication. Table 1 gives an overview over all random variables that were defined in this section.

Table 1

Overview over Defined Random Variables

Random variable	State	Meaning
$\delta \overset{obj}{\sim} Normal\left(\mu_\delta^{(o)}, \sigma_\delta^{2(o)}\right)$	objective	True effect size of the study
$e_i \overset{obj}{\sim} Bernoulli\left(\pi_e^{(o)}\right)$	objective	Fault indicator for the i-th code unit
$u_i \overset{obj}{\sim} Normal\left(\mu_u^{(o)}, \sigma_u^{2(o)}\right)$	objective	True error sizes associated with faulty code units
$e_i \overset{subj}{\sim} Bernoulli\left(\pi_e^{(s)}\right)$	subjective	Belief about the fault indicator for the i-th code unit
$\pi_e^{(s)} \overset{subj}{\sim} Beta\left(\alpha^{(s)}, \beta^{(s)}\right)$	subjective	Hyperprior for the belief of a fault occurring in a code unit
$\delta \overset{subj}{\sim} N\left(\mu_\delta^{(s)}, \sigma_\delta^{2(s)}\right)$	subjective	Belief about the true effect size of the study
$u_i \overset{subj}{\sim} N\left(\mu_u^{(s)}, \sigma_u^{2(s)}\right)$	subjective	Belief about the error sizes associated with faulty code units

Presenting `scerrModel` – an implementation in R

In this section, the **`scerrModel`** package for **R** 4.4.1 (R Core Team, 2024) will be presented. It offers a full implementation of the specified model and allows the user to

simulate a complete dataset of literature. It possesses easy-to-use functions to define agents and studies that allow building up the input for the simulation, consisting of all model parameters. Simulation results can then be analysed using a summarisation function as well as multiple pre-defined analysis plots for single studies or a subset of studies from the whole literature dataset. All functions return common **R** data structures that permit easy interfacing with other **R** packages and functionalities. The package uses defensive programming, relying on the **checkmate** package (Lang, 2017) for versatile argument checking. Analysis plots are generated using **ggplot2** (Wickham, 2016) and **patchwork** (Pedersen, 2024), allowing for easy modification by the user.

The package can be found on GitHub (https://github.com/advieser/scerrModel), and can be installed using the **remotes** package (Csárdi et al., 2024):

```
install.packages("remotes")
remotes::install_github("https://github.com/advieser/scerrModel")
```

Functionalities and Standard Workflow

At the beginning of each simulation, a set of agents and a set of studies must be defined. Combined, an agent and a study contain all necessary input parameters for the simulation of one study. To define these, the functions **create_agents()** and **create_studies()** should be employed which return tabular data. Table 2 lists all arguments of these functions and in doing so gives an overview over all input parameters for the simulation.

Table 2

Arguments to Creation Functions for Agents and Studies

Function	Argument	Explanation
`create_agents()`	`agent_id`	Unique identifier of an agent, necessary for assignment to a study.
	`subj_effect_mu`	Expected value $\mu_\delta^{(s)}$ of the agent's belief distribution for the true effect size δ
	`subj_effect_sigma`	Standard deviation $\sigma_\delta^{2(s)}$ of the agent's belief distribution for the true effect size δ
	`subj_prob_fault_alpha`	Shape parameter $\alpha^{(s)}$ for the Beta distribution for the probability of finding a fault within a code unit $\pi_e^{(s)}$
	`subj_prob_fault_beta`	Shape parameter $\beta^{(s)}$ for the Beta distribution for the probability of finding a fault within a code unit $\pi_e^{(s)}$
	`subj_error_size_mu`	Expected value $\mu_u^{(s)}$ of the agent's belief distribution for the error sizes u_i
	`subj_error_size_sigma`	Standard deviation $\sigma_u^{2(s)}$ of the agent's belief distribution for the error sizes u_i
`create_studies()`	`study_id`	Unique identifier for the study
	`agent_id`	Unique identifier of an agent, necessary for assignment to a study.
	`N`	Number of code units for that study
	`resources`	Initial amount of available resources for that study
	`cost`	Cost of performing a search round for that study
	`benefit`	Benefit of finding a fault within a search round for that study
	`obj_effect_mu`	Expected value $\mu_\delta^{(o)}$ of the true probability distribution for the true effect size δ
	`obj_effect_sigma`	Standard deviation $\sigma_\delta^{2(o)}$ of true probability distribution for the true effect size δ
	`obj_prob_fault`	Probability $\pi_e^{(o)}$ of finding a fault within a search round, parameter for the true distribution of fault indicators e_i
	`obj_error_size_mu`	Expected value $\mu_u^{(o)}$ of the true probability distribution for the error sizes u_i
	`obj_error_size_sigma`	Standard deviation $\sigma_u^{2(o)}$ of true probability distribution for the error sizes u_i

Once a set of agents and and a set of studies is defined, the user may combine the two by calling **`combine_agents_studies()`**.This is not necessary but can be useful to verify that each study has the intended agent assigned to it, or to get an overview over all input parameters for a study. For lack of a better term, the returned data frame can be considered a set of complete studies.

Having taken this preparatory steps, the user can now run the simulation by calling **`simulate_literature()`**. This takes either a data.frame of complete studies or the two separate data frames for agents and studies as input, that will be combined by simply calling **`combine_agents_studies()`** internally. The function returns a list, the first entry of which is the data frame of complete studies, the second and third entry describe the seed options that were set, and all following entries are lists (one per simulated study) that contain all final and intermediate results of the simulation of the respective study. This output is not designed for easy analysis and can be overwhelming at first. For this reason, the usage of the analysis functionalities of this package is recommended. However, this way all information describing the simulation is available to the user. For its interpretation, the reader is referred to the documentation of **`simulate_literature()`**. Lastly, the simulation function can take two additional optional arguments, **`seed`** and **`keep_seed_const`**, that will be explained in more detail in the next section.

All analysis functions take the list returned by **`simulate_literature()`** as input, plus, in some cases, some additional arguments. First of all, there is **`simulation_summary()`**, which returns a data frame of key variables describing the result of the simulation with one row per study. These variables include the number of search rounds that were performed, the reason why the agent decided to stop searching, the number of faults in the study (total, discovered and undiscovered), the true effect size and the final effect size observed in the round in that the agent decided to stop searching. When specifying the argument **`simple = FALSE`**, the return will also include all values contained in the set of complete studies, resulting in a data frame that contains the majority of

information about the final results of the simulations, allowing the user to easily run further analysis.

Aside from this summary function, the package offers several pre-defined plots for easy analysis of the simulation. For one, there are three functions to plot the development of one variable, respectively: the probability of finding a fault in the current search round (**`plot_prob_fault_this_round()`**), the percentage of the number of faults remaining relative to the number of code units remaining (**`plot_undiscovered_faults()`**), and the observed effect size in the current round (**`plot_obs_effect_sizes()`**). Each of these functions plots the curves for all simulated studies by default. However, a subset may be specified by passing a character vector to **`study_id`**. The function **`plot_obs_effect_sizes()`** can additionally be configured to show the true effect size of the respective study, and **`plot_prob_fault_this_round()`** may be configured to plot the cost-benefit-ratio of the respective study. All these functions may be plotted together directly, by calling **`plot_summary_panel()`**.Secondly, the function **`plot_distributions_heatmap()`** can be used to analyse the development of the updating process for a specific study.

Last of all, two functions exist to plot the distributions of remaining faults (**`plot_histogram_n_faults()`**) or the final effect sizes (**`plot_histogram_final_effect_sizes()`**) as histograms, across the whole literature (or a subset thereof). These enable analysis of the distribution of essential outcome variables in the literature.

Options for Seed Setting

The simulation results necessarily contain randomness, as the true effect size, the fault indicators and the error sizes in the objective reality are drawn randomly from their respective distributions. To still ensure reproducibility of results, the user has two different options for setting a seed which will be compared in the following paragraphs.

The first option is to pass a seed to the **`seed`** argument of **`simulate_literature()`** while keeping **`use_same_seed = FALSE`** (which is its default value). This is equivalent to running **`set.seed()`** before calling the simulation function. However, it avoids modifying the global environment and saves the seed in the returned list. The other option is to set **`use_same_seed = TRUE`** which leads to the function setting the seed given in **`seed`** before each generation of the objective reality for every study. This enables the user to analyse differences alternative choices of model parameters might make while negating the influence of randomness. However, it should be highlighted, that the simulated studies will be stochastically dependent and that this functionality is only intended for the analysis of the model behavior.

Example Simulation of One Literature Dataset

Now, an example simulation of ten studies will be performed to showcase the typical workflow when working with this package. A reproducible script for this example simulation can be found in **`analysis/example.R`** in the package repository on GitHub.

We first have to define our agents using **`create_agents()`**. For this example, we decide to give the agents different beliefs about the effect size they expect, while keeping the variance for this belief and all other parameters constant. We set $\alpha^{(s)} = 1$ and $\beta^{(s)} = 1$ which leads to the beta distribution taking the shape of a uniform distribution on the interval from zero to one, meaning the agents assume all probabilities of finding a fault in a code unit to be equally probable.

```
agents <- create_agents(
  agent_id = paste0("A", 1:10),
  subj_effect_mu = seq(0.2, 0.75, length.out = 10),
  subj_effect_sigma = 0.4,
  subj_prob_fault_alpha = 1,
  subj_prob_fault_beta = 1,
  subj_error_size_mu = 0,
  subj_error_size_sigma = 0.5
)
```

```
> str(head(agents))
'data.frame':     6 obs. of  7 variables:
 $ agent_id             : chr  "A1" "A2" "A3" "A4" ...
 $ subj_effect_mu       : num  0.2 0.261 0.322 0.383 0.444 ...
 $ subj_effect_sigma    : num  0.4 0.4 0.4 0.4 0.4 0.4
 $ subj_prob_fault_alpha: num  1 1 1 1 1 1
 $ subj_prob_fault_beta : num  1 1 1 1 1 1
 $ subj_error_size_mu   : num  0 0 0 0 0 0
 $ subj_error_size_sigma: num  0.5 0.5 0.5 0.5 0.5 0.5
```

Note that **`create_agents()`** allows for inputs of length one or of length of the **`agent_id`** input. Inputs of length one get automatically recycled to be the same length as the **`agent_id`** input.

We will now have to define a set of studies using **`create_studies()`**. For this example, we assume that each agent will perform the analysis for the same study. Consequently, we define ten studies that all have the same parameters except that we assign a different agent to each study.

```
studies <- create_studies(
  study_id = paste0("S", 1:10),
  agent_id = paste0("A", 1:10),
  N = 15,
  resources = 100,
  cost = 5,
  benefit = 10,
  obj_effect_mu = 1.0,
  obj_effect_sigma = 0.5,
  obj_prob_fault = 0.4,
  obj_error_size_mu = 0,
  obj_error_size_sigma = 0.6
)
```

```
> str(head(studies))
'data.frame':     6 obs. of  11 variables:
 $ study_id             : chr  "S1" "S2" "S3" "S4" ...
 $ agent_id             : chr  "A1" "A2" "A3" "A4" ...
 $ N                    : num  15 15 15 15 15 15
 $ resources            : num  100 100 100 100 100 100
 $ cost                 : num  5 5 5 5 5 5
 $ benefit              : num  10 10 10 10 10 10
 $ obj_effect_mu        : num  1 1 1 1 1 1
 $ obj_effect_sigma     : num  0.5 0.5 0.5 0.5 0.5 0.5
 $ obj_prob_fault       : num  0.4 0.4 0.4 0.4 0.4 0.4
 $ obj_error_size_mu    : num  0 0 0 0 0 0
 $ obj_error_size_sigma: num  0.6 0.6 0.6 0.6 0.6 0.6
```

We can now combine the two data frames and verify that the combination suits our needs.

```
cs <- combine_agents_studies(agents, studies)
```

```
> str(head(cs))
'data.frame':     6 obs. of  17 variables:
 $ agent_id              : chr  "A1" "A10" "A2" "A3" ...
 $ study_id              : chr  "S1" "S10" "S2" "S3" ...
 $ N                     : num  15 15 15 15 15 15
 $ resources             : num  100 100 100 100 100 100
 $ cost                  : num  5 5 5 5 5 5
 $ benefit               : num  10 10 10 10 10 10
 $ obj_effect_mu         : num  1 1 1 1 1 1
 $ obj_effect_sigma      : num  0.5 0.5 0.5 0.5 0.5 0.5
 $ obj_prob_fault        : num  0.4 0.4 0.4 0.4 0.4 0.4
 $ obj_error_size_mu     : num  0 0 0 0 0 0
 $ obj_error_size_sigma  : num  0.6 0.6 0.6 0.6 0.6 0.6
 $ subj_effect_mu        : num  0.2 0.75 0.261 0.322 0.383 ...
 $ subj_effect_sigma     : num  0.4 0.4 0.4 0.4 0.4 0.4
 $ subj_prob_fault_alpha: num  1 1 1 1 1 1
 $ subj_prob_fault_beta : num  1 1 1 1 1 1
 $ subj_error_size_mu    : num  0 0 0 0 0 0
 $ subj_error_size_sigma: num  0.5 0.5 0.5 0.5 0.5 0.5
```

Now, we are ready to run the simulation using **`simulate_literature()`**. We pass the data frame of complete studies to the function and decide to set a seed for reproducibility purposes. Afterwards, we take a quick look at the raw output. Here data frame of complete studies and all simulation results except for those of the first study are omitted.

```
lit <- simulate_literature(complete_studies = cs, seed = 1234L)
```

```
> str(lit)
List of 13
 $ complete_studies:'data.frame':   10 obs. of  17 variables:
 $ seed            : int 1234
 $ use_same_seed   : logi FALSE
 $ S1              :List of 11
  ..$ obj_effect_size      : num 0.396
  ..$ fault_indicators     : int [1:15] 1 1 1 1 0 0 1 0 1 0 ...
  ..$ error_sizes          : num [1:15] -0.373 -0.439 -0.31 -1.05 0.528
...
  ..$ observed_effect_sizes: num [1:16] -2.071 -1.697 -1.258 -0.948 0.102
...
  ..$ p_after_fault_ind    : num [1:16, 1:16] 6.25e-02 1.11e-04 1.17e-05
1.02e-05 1.68e-05 ...
  ..$ p_after_effect       : num [1:15, 1:16] 9.66e-08 1.21e-08 9.38e-08
8.60e-07 6.30e-05 ...
  ..$ stopped_in_round     : num 16
  ..$ stopping_reason      : chr "Search completed"
  ..$ p_fault_this_round   : num [1:15] 0.668 0.754 0.792 0.814 0.818 ...
  ..$ eu_criterion         : num [1:15] 1.34 1.51 1.58 1.63 1.64 ...
  ..$ remaining_resources  : num 25
 $ S10             :List of 11
 $ S2              :List of 11
 $ S3              :List of 11
 $ S4              :List of 11
 $ S5              :List of 11
 $ S6              :List of 11
 $ S7              :List of 11
 $ S8              :List of 11
 $ S9              :List of 11
```

Since it is rather difficult to make any sort of inferences from this alone, we should use the analysis functions the package provides. First off, we want to get a better overview over the results using the **`simulation_summary()`** function.

```
result <- simulation_summary(lit)
> str(head(result))
'data.frame':     6 obs. of  15 variables:
 $ study_id                 : chr  "S1" "S10" "S2" "S3" ...
 $ agent_id                 : chr  "A1" "A10" "A2" "A3" ...
 $ N                        : num  15 15 15 15 15 15
 $ resources_before         : num  100 100 100 100 100 100
 $ resources_after          : num  25 100 100 25 25 95
 $ cost                     : num  5 5 5 5 5 5
 $ benefit                  : num  10 10 10 10 10 10
 $ bcr                      : num  2 2 2 2 2 2
 $ n_rounds                 : num  16 1 1 16 16 2
 $ reason                   : chr  "Search completed" "Expected utility
too low" "Expected utility too low" "Search completed" ...
 $ n_faults_total           : num  8 5 2 7 11 6
 $ n_faults_discovered      : num  8 0 0 7 11 0
 $ n_faults_undiscovered    : num  0 4 2 0 0 6
 $ real_effect_size         : num  0.396 0.981 0.374 2.068 0.752 ...
 $ final_observed_effect_size: num  0.3965 0.0431 0.6962 2.0677 0.7522 ...
```

There are several columns that are of particular interest to us. The first of which is **n_rounds** which gives the number of search rounds that were performed.

```
> result$n_rounds
 [1] 16  1  1 16 16  2 16 16 16  1
```

Here, a value of 16 (or generally of $N + 1$) indicates that the agent fully checked all 15 code units. The plus one is there to indicate that the agent did not stop at the beginning of the 15-th round before checking the final code unit. We can see that for the most part agents either did not begin any fault searching behaviour or checked the whole code for faults. This is a typical behaviour for certain combinations of parameters and will be discussed more in the next section.

Another column of interest is the reason why the agent stopped. This can either be due to the subjective expected utility being too low, due to the resources being depleted, or due to the agent having checked all code units.

```
> table(result$reason)
Expected utility too low         Search completed
                       4                        6
```

We can see that in the cases where the agent decided to stop, they did so because of a too low subjective expected utility. In this example, we could have derived as much from the number of rounds, as we knew that there were enough resources for a full examination of the code. However, in simulations that show a more diverse set of stopping behaviours, this column can be very informative.

We can also look at the columns describing the number of faults: **n_faults_total**, **n_faults_discovered**, **n_faults_undiscovered**.

```
> result$n_faults_total
 [1]  8  5  2  7 11  6  9  5  7  6
```

The total number of faults simply follow from random sampling from a Binomial distribution that has the parameters $n = N$ and $\pi = \pi_e^{(o)}$. From the previous outputs we already know that in six of the studies all faults were discovered since the agents checked all code units,

and in the three studies no faults were discovered since the agent performed no fault searching whatsoever. In study S5 where the agent searched for a fault once, they found not fault, as can be seen below:

```
> result$n_faults_discovered[result$study_id == "S5"]
[1] 0
```

Again, in simulation with a more diverse set of stopping time points, this information may be more interesting. For instance, it allows us to plot the distribution of the number of remaining faults in a literature dataset (see later in this section).

Lastly, we can take a look at the **`real_effect_size`** and the **`final_observed_effect_size`**.

```
> result$real_effect_size
 [1] 0.3964671 0.9811849 0.3740071 2.0676913 0.7522083 0.6006651 1.6975739
1.7039221 1.3520901 0.3738851
> result$final_observed_effect_size
 [1]  0.39646713  0.04305447  0.69624986  2.06769131  0.75220828 -
0.91935969  1.69757395  1.70392206  1.35209009  0.63428412
```

If we look at the studies where the agent performed no search round, we can see that the final (and at the same time initial) observed effect sizes were not unrealistic. This may also play a role in why the agents decided to not search for faults.

```
> result$final_observed_effect_size[result$n_rounds == 1]
[1] 0.04305447 0.69624986 0.63428412
```

From these columns we may also calculate the total error left in a study at publication.

```
> result$final_observed_effect_size - result$real_effect_size
 [1]  0.0000000 -0.9381304  0.3222428  0.0000000  0.0000000 -1.5200248
0.0000000  0.0000000  0.0000000  0.2603990
```

Having got a first overview over the simulation results, we may want to explore the results visually using the package's plotting capabilities to gain more insights.First, we get a first overview using the **`plot_summary_panel()`** function which builds a panel of three plots (**`plot_prob_fault_this_round()`**, **`plot_undiscovered_faults()`**, **`plot_obs_effect_sizes()`**). The result can be seen in Figure 1.

```
plot_summary_panel(lit, cbr = TRUE)
```

In the first plot, we can see how the belief of an agent for a fault occurring in the next code unit developed over the analysis of the study. The dashed line indicates the cost-benefit-ratio as the threshold for the decision to search. As this ratio is equal for all studies only one such line can be seen. The second plot shows the development of the number of undiscovered faults in relation to the number of code units that remain unchecked in that round. Noticeably, this is difficult to interpret visually with so many studies being plotted. For this reason, one might want to take a closer look at separate subset of the studies by specifying **`study_id`** when creating this plot. The last plot shows the development of the observed effect sizes across rounds. It can be noted that these values go up and down since the error sizes were drawn from a distribution centred around zero.

We may also take a closer look at individual studies and examine the development of the belief distributions over the number of remaining faults. Here, we choose to view the distributions after the effect sizes and before the fault indicators were observed. The result can be seen in Figure 2.

```
plot_distributions_heatmap(lit, study_id = "S7", type = "after_effect")
```

Figure 1

A Panel of Summary Plots Describing the Example Simulation

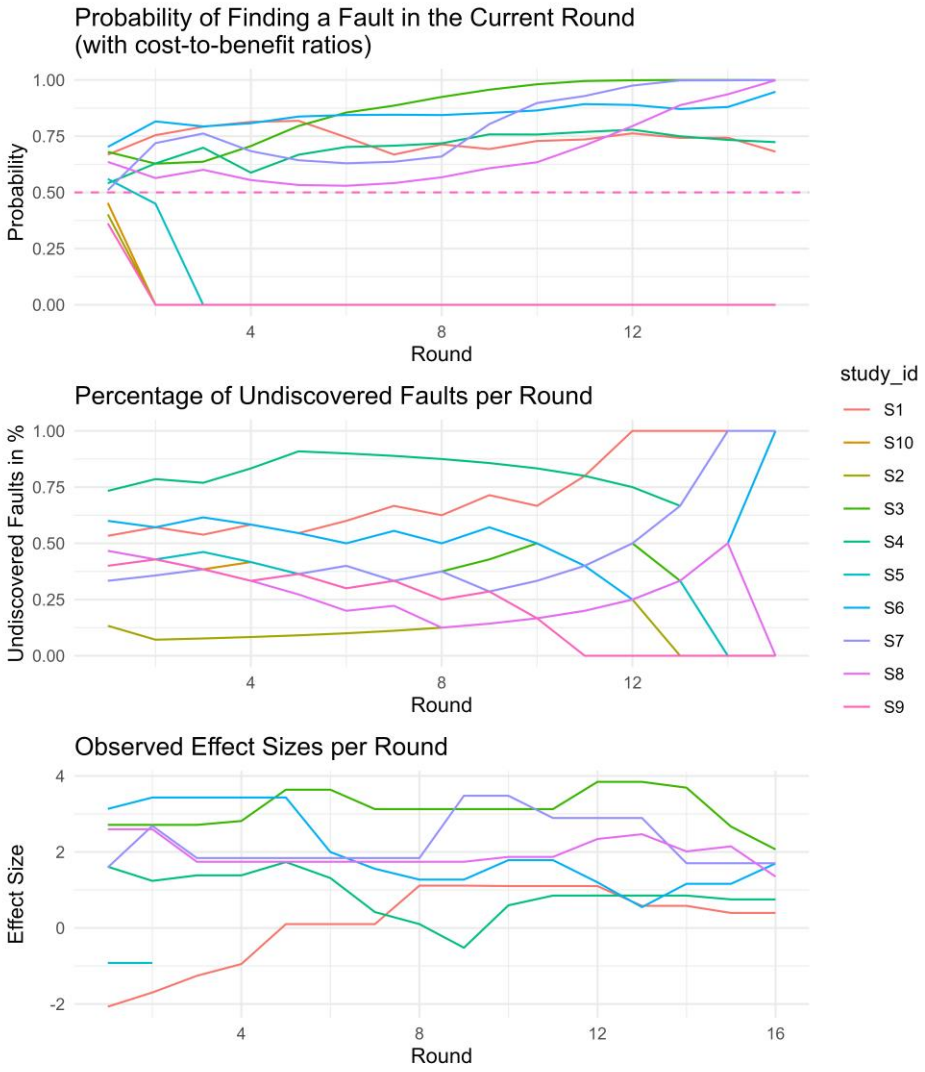
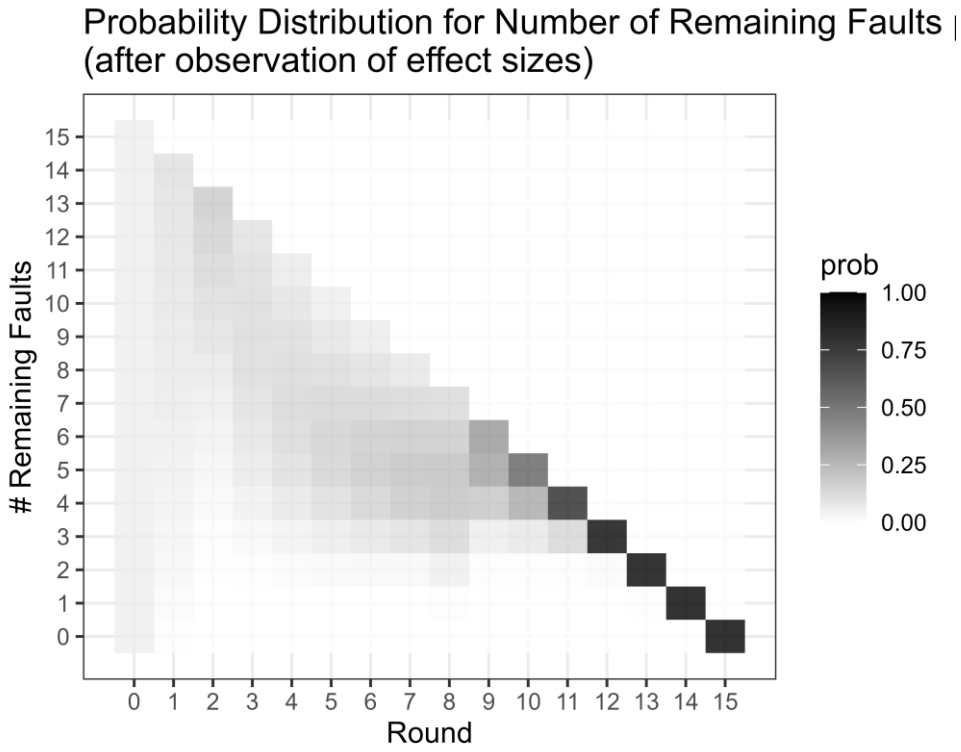


Figure 2

Distributions over the Number of Remaining Faults in Study S7 per Round



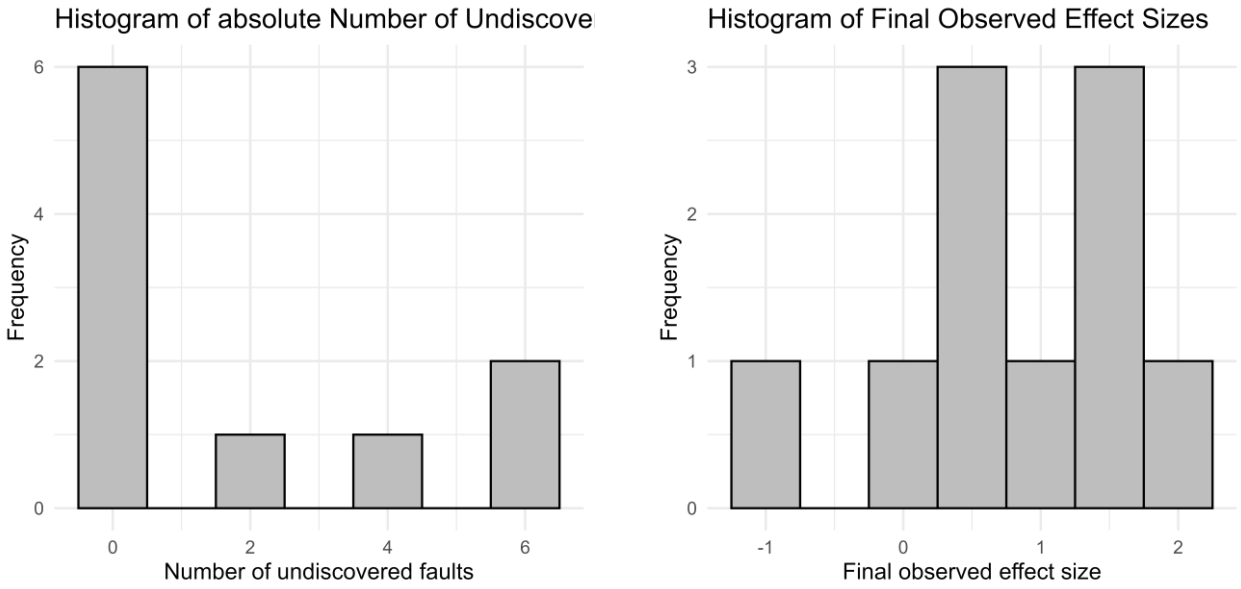
We can see the initial belief in round zero (i.e. before the first round) reflects the uniform distributions we specified as a prior. After this, the agent first tends to assuming a larger number of faults remaining before concluding near the end that as many faults as the number of code units they have not checked yet remain. This last conclusion is a general pattern that can be observed and will be discussed in the next section.

Finally, we can investigate the distributions over the number of remaining faults as well as the effect sizes in a published study across our simulated literature.

```
plot_histogram_final_effect_sizes(lit, binwidth = 0.5)
plot_histogram_n_faults(lit, type = "absolute")
```

Figure 3

Histograms for Distributions of Published (Simulated) Literature



Analysis of Model Behaviour and Sensitivity Analysis

In this section, general tendencies and patterns in the model behaviour, which stood out during testing, are discussed. The first of which is the tendency of many agents, under a seemingly wide variation of model parameters, to never look for any faults. On the other hand, there are also a set of combinations of model parameters that lead to agents always checking the full source code. The conditions of these occurrences will first be derived theoretically, and a short sensitivity analysis is performed to check the validity of the conclusions. Secondly, the relationship between the number of code units, fault indicators and error sizes will be discussed. Lastly, in studies where the agent performs multiple search rounds, a pattern for the distribution of remaining faults became apparent. This has been noticed previously by F. Melinscak. Possible reasons for this will be discussed.

Determinants of Decision to Stop

As noted above, under a large set of conditions, the agents choose to either never start a fault search or perform an extensive fault search across of all code units in a high

majority of cases. Both types of occurrences seem unlikely. If the model is intended to simulate researchers' real behaviour, it should be calibrated in a way to reflect the likely reality that researchers would possibly stop after examining any number of code units. Therefore, it would be beneficial to be able to better describe the conditions under which these behaviours occur.

Benefit, Cost and Resources

As described in the presentation of the specification, an agent decides to start another search if (a) more resources are available than a search would cost, and (b) if the decision criterion for the subjective expected utility is met. This was formulated as

$$P(e_i = 1|O_i) \cdot \frac{benefit}{cost} > 1\,.$$

This shows that the benefit-cost-ratio, which is constant for a study, is a major determinant for the decisional behaviour. However, if this formula is restated as

$$P(e_i = 1|O_i) > \frac{cost}{benefit}$$

a direct interpretation of the cost-benefit-ratio as the probability threshold—below which the subjective probability of finding a fault in the next search round must not fall— becomes apparent. This allows the modeller to easily predict, based on the cost and benefit, whether the agent will never initiate or always complete the search—provided sufficient resources are available—since either a very high or very low threshold is implied by the cost and benefit values. Lastly, to conclude the discussion on the role of costs, benefits, and resources, the modeller can also easily calculate the maximum number of search rounds an agent could perform, given the cost of each search round and the amount of initial resources available.

Subjective Probability of Finding a Fault in the First Round

The other obvious determinant of the decisional behaviour is the belief of the agent about finding a fault in the next search round, i.e. $P(e_i = 1|O_i)$. For later search rounds, the influences on this probability are hard to describe, since by then the model behaviour would

be too complex to easily retrace. However, for the first search round this is possible. This allows us to investigate what initial conditions would lead to the agent never starting the search.

For the first round, we are effectively considering

$$P(e_1 = 1|d_1) = \frac{E(c_1|d_1)}{N}.$$

which means that the distribution over c_1 given d_1 is of primary interest to us. This is given by the first updating process

$$P(c_1|d_1) \propto P(d_1|c_1) \cdot P(c_1)$$

From this, we can see that the prior for the number of remaining faults c_1, which is defined through the Beta-Binomial distribution, and that the likelihood of the first observed effect size d_1 influence the belief about finding a fault in the first round. We can further note that, for searching to become the more likely choice, the expected value of this distribution needs to increase, implying that the agent would anticipate finding more faults. Consequently, the probability mass of the distribution must shift to the right. This shift occurs when both the likelihood function and the prior assign higher likelihood or probability, respectively, to the values further to the right. Given that the likelihood and prior interact multiplicatively, the effect is maximized when both exhibit this behaviour. We will now examine the likelihood function and the prior separately to determine the conditions under which this scenario occurs.

Influence of the Choice of Hyperparameters for the Prior. We will start by examining the prior for the number of remaining faults c_1. The prior is defined through the parameters N, $\alpha^{(s)}$, and $\beta^{(s)}$. It can be remarked that the prior is most important in the first search round, as its role is that of an initial belief. This belief is updated throughout the entire process, and as the process continues, the initial prior becomes less influential. As mentioned earlier, for searching to become the more probable choice, the prior must assign a higher probability to the scenario where a larger number of faults remain.

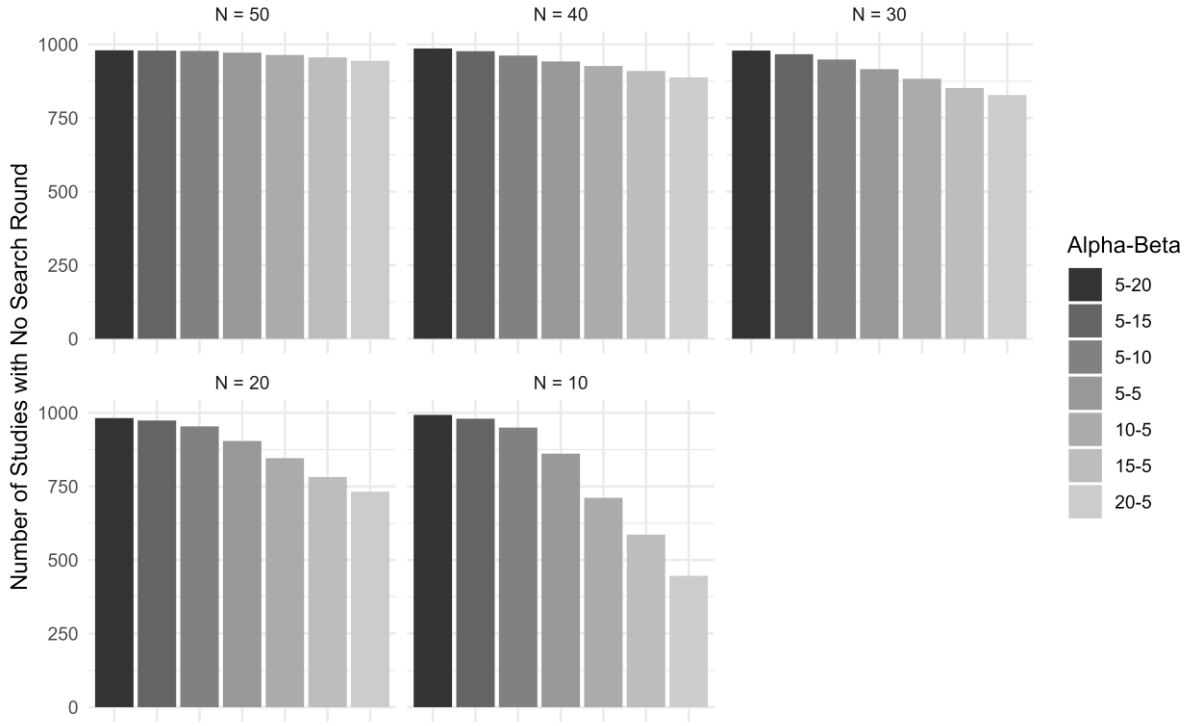
The prior follows a Beta Binomial distribution where the Beta hyperprior models the agent's initial belief about the probability of a fault occurring in a code unit. If the agent would assume this probability to be higher, the Beta distribution would indicate this by assigning more probability to values closer to one. This is generally the case when $\alpha^{(s)}$ is larger than $\beta^{(s)}$.

To verify this conclusion, a sensitivity analysis was performed. To do this, seven different combinations of $\alpha^{(s)}$ (5, 5, 5, 5, 10, 15, and 20) and $\beta^{(s)}$ (20, 15, 10, 5, 5, 5, 5) were used to simulate 1000 studies each. This was then done for five different values of N (10, 20, 30, 40, and 50) to investigate the interaction effect between these parameters. All other parameters were kept constant. A reproducible script of this analysis can be found in **`analysis/hyperprior_analysis.R`** in the package repository.

For analysis, the percentage of studies that passed the first round for each unique combination of the three parameters was examined. As can be seen in Figure 4, a general pattern occurred that confirms the above derivation: For hyperparameters that indicate a higher belief that a fault could occur in a code unit, the number of studies that passed the first search round increases. Furthermore, this increase gets stronger with a smaller number of code units N. This is because increasing N, increases the variance of the Beta-Binomial prior. In this simulation, this leads to the prior having a smaller impact on the posterior. It should however be kept in mind, that this analysis doesn't allow any statement to be made about the generalisability of this relationship to other combinations of parameters that were kept constant here, especially those influencing the likelihood.

Figure 4

Sensitivity Analysis for Variation of $\alpha^{(s)}$, $\beta^{(s)}$, and N



Influence of Choice of Parameters Affecting the Likelihood. The first thing that can be noted about the likelihood is that all subjective parameters that describe the agent's belief about the true effect size and the error sizes influence it, since they are used to define the subjective belief about the number of remaining faults. Secondly, we can note that the same is true for all parameters describing the objective reality, since they control the calculation of the first observed effect size. At the same time, the objective parameters are all parameters that influence the random processes that are part of the simulation: the sampling of a true effect size, the sampling of fault indicators as well as the sampling of error sizes. This is the reason why we must expect that the decisional behaviour in the first round will, to an extent, depend on randomness.

For the likelihood function to assign higher likelihoods to a greater number of remaining faults, the observed effect size would need to be as close as possible to $\mu_{\delta}^{(s)} + N \cdot \mu_{u}^{(s)}$, as this would represent the rightmost value in the domain of the likelihood function.

Conceptually, this means that the observed effect size should be near values that the agent would find more likely if a significant number of faults were still present in the code. However, if the variance in the belief distribution about error sizes is higher, the distance between the observed effect size and the expected value of the likelihood may be larger. This increased variance would result in a higher likelihood being assigned to the observed effect size compared to when the agent assumes a smaller variance.

Sadly, a sensitivity analysis for possible combinations of these parameters, to confirm this derivation, is not feasible due to the combinatoric level of complexity introduced by the number of influencing parameters.

Summary. In conclusion, all parameters affect the decisional behaviour in the first round. This makes it difficult to formulate definite facts about all possible ways this behaviour could be influenced. We may however note a couple of general tendencies.

First of which is the role of resources, cost and benefit. This relationship is the exemption to the above sentence, in that the impact of these parameters can be clearly described. The cost-benefit-ratio formulates a definite threshold for the agent's belief about finding a fault. Should it not be exceeded in a search round, the agent stops all further searching. Additionally, the cost of performing a search round in relation to the initial resources available gives an upper boundary for the number of search rounds that are possible.

Secondly, the subjective belief of finding a fault in the first round determines the decisional behaviour. This is influenced by all model parameters. Since this includes the parameters defining the distributions of the true effect size, the fault indicators, and the error sizes from which we randomly sample, some randomness is at play here. Aside from that, searching can become more probable depending on the hyperparameters describing the Beta-Binomial prior, for which a relatively clear relationship could be shown, and on the parameters influencing the likelihood of the initial observed effect size. This however seems to be a more complex interaction.

Having determined these influencing factors, some directions for the calibration of parameters to allow a more diverse model behaviour have been discovered. However, ultimately, it would not be clear changing which these parameters would result in the best, or most realistic, model behaviour since empirical data about the input parameters (e.g. resources, cost, and benefit) is not available. This will be discussed more in the next section.

While the short sensitivity analysis performed above focused on the conditions under which an agent even starts to search for faults, the present model allows for more far-reaching conclusions about the general state of the literature to be made (should the model be accurate enough). For instance, the effect of parameter settings on the final distributions of total errors of effect sizes or the number of faults remaining in studies post publication would be of further interest. However, for this to be possible, it would first be beneficial to find those values in the parameter space that lead to a more diverse set of model behaviours which was why the above analysis focused on said behaviour tendencies.

Relation between Number of Code Units, Probability of Faults, and Error Sizes

As derived above, the likelihood of the observed effect size in a round plays an important role in determining the agent's decision for whether to perform a search or not. As described early in the description of the specification, the observed effect size is calculated using the true effect size, the fault indicators and the error size. Here exists an important relationship between the later two and the number of code units.

This can be explained quite directly: If the number of code units in a study is higher than in another, more fault indicators get drawn. These fault indicators get multiplicated with the error sizes to compute the total error. However, if neither the probability of a fault occurring ($\pi_e^{(o)}$) nor the error sizes (determined by $\mu_u^{(o)}$ and $\sigma_u^{2\,(o)}$) are modified, the error will be much higher in the study with more code units. This generally makes sense since there are more opportunities for the agent to make mistakes when writing the code. However, in

the stage of calibrating the present model, this relationship should be kept in mind, since it may influence the model behaviour significantly.

Pattern in Belief about the Number of Remaining Faults

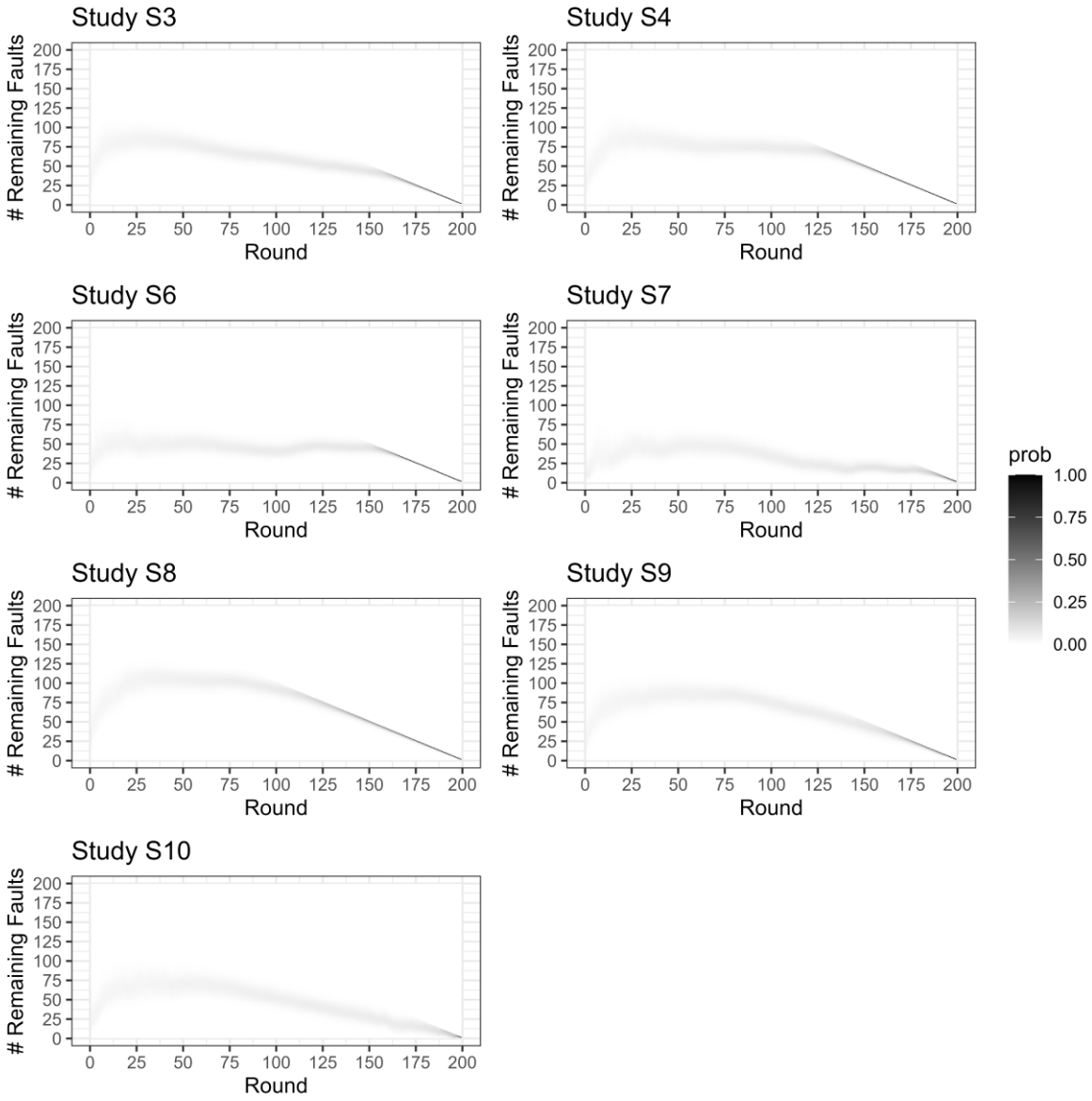
A particular pattern has been noted in the development of the belief distribution about the number of remaining faults, i.e. the distribution that gets updated across search rounds. The pattern is, that for larger N, there is a tendency for the probability mass to become concentrated on exactly one value. The agent seems to strongly believe that as many faults remain in the code as code units remain. This tendency can already be seen for a smaller N in Figure 2. This was first observed by F. Melinscak.

In this section, I will report my attempts at reproducing this behaviour before listing possible reasons for this pattern. For this, a simulation of ten different studies was run with largely differing parameters. The number of code units was however kept constant at $N = 200$. Furthermore, parameter values were chosen to make reaching later search rounds more probable, to facilitate finding the behavioural pattern. This included setting a very high benefit-cost-ratio as well as setting the hyperparameters of the prior to values supporting this goal. The reader may be referred to the reproducible script of this analysis in the **`analysis/pattern_analysis.R`** in the package repository to see the used parameter values.

Of the ten studies, seven (nearly) finished searching all code units. These were then extracted and their distributions for the number of remaining faults per round (after observing the computed effect size in that round) are plotted in Figure 5.

Figure 5

Belief Distributions of Remaining Faults for Large N in Seven Simulated Studies



Generally, the described pattern can be observed in nearly all plotted studies. However, what is particularly interesting is that it appears to set in at vastly different time points. While it can be clearly seen in Study S8, where it begins near round 120, in Study S10 it seems to occur near the very end where one could even doubt whether it is the same behaviour.

This raises the question whether the pattern emerges due to a fault in the specification or implementation of the model, or whether it is a genuine consequence of the assumptions that were made. In the following paragraphs, possible reasons underlying the occurrence of this pattern are discussed.

For one, this could be due to a possible faulty implementation of the specification. While the specification assumes that the likelihood in the second application of Bayes' rule (after the fault indicator was observed) to come from a Bernoulli distribution with parameter $\pi = \frac{c_i}{N-(i-1)}$, the original draft of the specification by F. Melinscak calculates the likelihood for parameter values of $\pi = \frac{c_i}{N}$ where $c_i = 1, \ldots, N$. As this draft is the basis for the present implementation, this is also the case for the present implementation. However, the impact of this difference is difficult to ascertain theoretically.

Conceptually related to this is the second possible reason for this pattern: The assumption about the distribution for this likelihood, where conditional independence is assumed, i.e. $P(e_i = 1|c_i, O_i) = P(e_i = 1|c_i)$. This assumption has not yet been verified, and it is possible that the likelihood does not actually follow a Bernoulli distribution at all. However, since this step has not yet been implemented as originally specified, the implementation would have to be corrected first before investigating this potential explanation further.

Lastly, the behaviour pattern could be a consequence of the distributional assumptions that were made. At this point, the pattern could be considered a outcome result of the model.

Discussion

In the present thesis, a formal model for the emergence of scientific errors in data analysis of empirical research and its specification have been reviewed, an implementation of that model has been introduced, and a non-exhaustive analysis of model behaviour has been performed. The following discussion will focus on critically reviewing the current state

of the model, its specification, and the implementation presented here. This will include open questions concerning the model and its assumptions, directions and difficulties for a possible empirical verification of the model, as well as possible improvements for the implementation.

Open Questions Concerning Model Assumptions

Defining Code Units

The concept of code units, that is introduced for the specification of the model, has led to some internal discussion. While being a fundamental element for all following steps, it remains challenging to concretise, which is why its definition was intentionally left abstract. Potential analogues in reality might include lines of codes, function calls with their respective argument specifications (where one code unit would correspond to one argument of the function), or analytic transformations within the functions invokied during the analysis.

A problem with these suggested analogues is the vastly different scale of the number of code units that each definition would imply for the same source code. Additionally, depending on how code units are defined, the assumption of an equal probability of faults occurring in each code unit might vary in its realism. For instance, one would not expect the same fault occurrence probability for lines of codes, that may be of different lengths, may contain a different number of variables that are being referred, etc.

As we are modelling faults that were introduced by mistakes made by researchers when specifying their analysis plan (e.g. as code), the definition of a code unit could possibly correspond to the points where a researcher might make a mistake when specifying their analysis plan, e.g. when specifying variable names or performing indexing operations.

Finally, if this model were to be used to simulate literature based on empirical calibration (see later), the concept of a code unit would need to be more clearly defined. A precise definition is essential to accurately determine the number of code units in a real study, given the source code of the analysis performed in that study.

Distributional Assumptions

For the specification of the formal model, several distributional assumptions were made, about the objective reality as well as the possible beliefs of agents. Ideally, each of these assumptions could be justified by either theoretical or empirical considerations. At the present stage, many of these justifications are missing.

For the objective reality, the true effect sizes δ are assumed to be normally distributed. It is difficult to judge whether this can be justified. However, it can be remarked that a justification through empirical data would not be advisable as estimates for the distribution of effect sizes that are based on published literature will almost surely be biased due to publication bias. The assumption of a Bernoulli distribution for the occurrence of faults within code seems reasonable, as this distributional family is often used for fault quantities. However, the distributional assumption for error sizes presents a significant challenge. As stated earlier, there is no a priori reason to justify this assumption, which poses a considerable threat to the model's validity. However, assumptions made about the objective reality have the advantage of being flexible. If a different distribution is assumed for any of these processes, it would primarily impact the randomness-based calculation of the observed effect sizes.

However, as subjective beliefs are supposed to be approximations of the data generating processes in reality, they are generally modelled to come from the same distributional families as their analogues in the objective reality. This is more problematic since changing the assumptions about the distributional families for the subjective beliefs would affect the Bayesian updating process. Here, the current implementation relies on these distributional families to enable the use of an analytical solution for the applications of Bayes' rule, instead of computationally intensive numerical solutions. If the distributional assumptions cannot be upheld going forward, this might necessitate switching to using numerical methods.

Overall, the distributional assumptions made for the specification are quite strong, which may potentially limit the utility of the model, as inferences drawn from simulations

would depend heavily on the validity of these assumptions. A related challenge is the verification of these assumptions, which adds an additional layer of complexity to the model's application and interpretation.

Directions and Difficulties for Empirical Verification

The preceding sections have already emphasized the critical role of the assumptions that underpin the model specification. A possible solution to the problem of justifying these assumptions could be empirical calibration of the model parameters. Ideally, a dataset would exist that would allow estimation of the model parameters and that would also contain the necessary information for verification of the accuracy of the simulated results. With this a hard empirical test for the validity of the model would be possible.

However, it is unlikely that such a dataset could be compiled. This is partly due to the sheer number of model parameters, as well as the abstractness of these. Additionally, some parameters will most likely not be able to be operationalised. This could include the number of code units N as well as the currently very abstract concepts of resources, benefit of finding a fault and costs, especially in relation to each other.

As an alternative, model parameters could be calibrated to ensure that the model predicts distributions similar to those found in datasets containing the number of faults in software or the error sizes in statistical analyses. While this approach constitutes a less stringent test of model validity, it may still provide valuable insights into the conditions under which faults are likely to persist until publication.

Improvements for the Implementation

Finally, weaknesses and possible improvements for the present implementation should be discussed. First, the possibly fault implementation of the specification for the likelihood in the second application of Bayes' rule in a round, should be investigated more closely. This is likely to change the model behaviour immensely, as it is a key part of the belief updating process.

Second, one unanticipated edge case was found during testing, and this should be addressed in future versions. The edge case arises if the likelihood calculation in one of the applications of Bayes' rule results in a zero likelihood for all parameter values. This can also occur if the likelihood is so small that it underflows to zero during computation. This will then lead to a division by zero when calculating the posterior, introducing `NaN`s. The simulation will raise an error, when the posterior is used to calculate the expected utility criterion.

Third, since the effect of different combinations of parameter values has not yet been exhaustively analysed, the examples in the documentation may utilise parameter values that result in simulations where no search rounds are conducted. This should be addressed in future revisions. Lastly, it would be beneficial to incorporate unit tests for key components of the simulation model to ensure robustness, accuracy, and reliability across a broader range of parameter configurations.

Conclusion

This thesis has reviewed and implemented a formal model by Kohrt et al. (2023) aimed at understanding the emergence, detection, and correction of errors in data analysis within empirical research. By focusing on errors that impact the numerical outcomes of statistical analyses, the model provides insights into how undetected faults can persist through the research process and ultimately influence published findings.

The implementation of this model in R, through the `scerrModel` package, allows for the simulation of these processes, enabling users to examine the dynamics of error propagation and the decision-making behaviour of agents responsible for fault detection, as postulated by the model. An initial non-exhaustive analysis of the model's behaviour using this implementation could only scrape at the surface due to its combinatoric complexity with a total of 15 input parameters. However, tendencies about the decisional behaviour of agents could be identified. Additionally, a pattern in the model behaviour for a large number of code units could be replicated.

Despite the potentially valuable insights provided by the model, several limitations must be acknowledged. The strong distributional assumptions necessary for the model's operation may not always align with real-world scenarios, potentially limiting its applicability. In addition, the abstract definition of "code units" and the challenges in empirically calibrating the model parameters present further areas of uncertainty. In the near future, further work should focus on analysing effects of different combinations of model parameters to allow calibration of the model, as well as finding justifications for the assumptions that are made to form the foundation for the model.

In conclusion, while the current model offers a promising approach to understanding scientific errors in data analysis, ongoing refinement and empirical testing are necessary to fully realize its potential and to contribute to the broader goal of improving the robustness and reliability of scientific research by furthering the understanding of the occurrence of scientific errors.

References

American Psychological Association (Ed.). (2020). *Publication manual of the American psychological association* (Seventh edition). American Psychological Association.

Andersen, L. E., & Wray, K. B. (2019). Detecting errors that result in retractions. *Social Studies of Science*, *49*(6), 942–954. https://doi.org/10.1177/0306312719872008

Borsboom, D., Van Der Maas, H. L. J., Dalege, J., Kievit, R. A., & Haig, B. D. (2021). Theory Construction Methodology: A Practical Framework for Building Theories in Psychology. *Perspectives on Psychological Science*, *16*(4), 756–766. https://doi.org/10.1177/1745691620969647

Csárdi, G., Hester, J., Wickham, H., Chang, W., Morgan, M., & Tenenbaum, D. (2024). *remotes: R package installation from remote repositories, including 'GitHub'* [Manual]. https://CRAN.R-project.org/package=remotes

International Organization for Standardization. (n.d.). *ISO/IEC/IEEE 24765:2017*. Retrieved 12 June 2024, from https://www.iso.org/standard/71952.html

Ioannidis, J. P. A. (2005). Why Most Published Research Findings Are False. *PLOS Medicine*, *2*(8), e124. https://doi.org/10.1371/journal.pmed.0020124

Karni, E. (2008). Savage's Subjective Expected Utility Model. In Palgrave Macmillan (Ed.), *The New Palgrave Dictionary of Economics* (pp. 1–5). Palgrave Macmillan UK. https://doi.org/10.1057/978-1-349-95121-5_2467-1

Kohrt, F., Melinščak, F., McElreath, R., & Schönbrodt, F. D. (2023). *A theoretical model of analysis errors and their correction in scientific research* (Version 1.0). Zenodo. https://doi.org/10.5281/ZENODO.10053574

Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t-tests and ANOVAs. *Frontiers in Psychology*, *4*, 863. https://doi.org/10.3389/fpsyg.2013.00863

Lang, M. (2017). checkmate: Fast Argument Checks for Defensive R Programming. *The R Journal*, *9*(1), 437–445.

Lindley, D. V. (2000). The Philosophy of Statistics. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *49*(3), 293–337. https://doi.org/10.1111/1467-9884.00238

Melinščak, F. (2023). *Error search decision-making model* [Personal communication].

Nakagawa, S., & Cuthill, I. C. (2007). Effect size, confidence interval and statistical significance: A practical guide for biologists. *Biological Reviews*, *82*(4), 591–605. https://doi.org/10.1111/j.1469-185X.2007.00027.x

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), aac4716. https://doi.org/10.1126/science.aac4716

Pedersen, T. L. (2024). *patchwork: The composer of plots* [Manual]. https://CRAN.R-project.org/package=patchwork

R Core Team. (2024). *R: a language and environment for statistical computing* [Manual]. R Foundation for Statistical Computing. https://www.R-project.org/

Savage, L. J. (1954). *The foundations of statistics* (1st ed.). Wiley.

van Dongen, N., van Bork, R., Finnemann, A., Haslbeck, J. M. B., van der Maas, H. L. J., Robinaugh, D. J., de Ron, J., Sprenger, J., & Borsboom, D. (2024). Productive explanation: A framework for evaluating explanations in psychological science. *Psychological Review*. https://doi.org/10.1037/rev0000479

van Ravenzwaaij, D., Bakker, M., Heesen, R., Romero, F., van Dongen, N., Crüwell, S., Field, S. M., Held, L., Munafò, M. R., Pittelkow, M. M., Tiokhin, L., Traag, V. A., van den Akker, O. R., van ‘t Veer, A. E., & Wagenmakers, E. J. (2023). Perspectives on scientific error. *Royal Society Open Science*, *10*(7), 230448. https://doi.org/10.1098/rsos.230448

Wickham, H. (2016). *ggplot2. Elegant Graphics for Data Analysis*. Springer International Publishing. https://doi.org/10.1007/978-3-319-24277-4

Eidesstattliche Erklärung

Hiermit versichere ich, Keno Mersmann, geboren am 12.07.2000, dass ich die vorliegende Arbeit selbstständig verfasst habe, dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, und dass ich diese Arbeit an keiner anderen Stelle eingereicht habe.

_____ _____

Ort, Datum Unterschrift