How Many Iterations are Enough? Bayesian Iterative Outlier Detection for Conservative Randomized Testing

Richard Carson

February 6, 2026

Abstract

It is a common problem in software testing to determine how many iterations of a randomized test are required to achieve a desired confidence level that edge cases have been adequately covered. Traditional approaches often rely on fixed iteration counts based on heuristic estimates of failure probabilities, which can lead to either insufficient testing or excessive computational costs, or on fuzzing techniques that may not be suitable for use in continuous integration (CI) pipelines due to their resource intensity.

Especially for numerical software, where failures may be caused by outliers in the data set, synthetic data with known outlier distributions are often used to evaluate robustness. However, determining the appropriate number of test iterations in such scenarios remains a challenge.

Our method bridges the gap between sequential analysis and Bayesian methods by providing a Bayesian iterative approach to outlier detection in randomized testing, allowing for adaptive determination of test duration based on observed failure rates and prior knowledge of outlier distributions. This approach not only enhances the reliability of test outcomes but also optimizes resource utilization by tailoring the number of test iterations to the specific characteristics of the data set and the testing environment.

This paper introduces the Bayesian Iterative Outlier Detection (BIOD) algorithm, and demonstrates that it can effectively balance computational cost with confidence in results, even when parameters are poorly specified. Empirical evaluation shows that BIOD accurately models failure rates under independent, normally distributed errors, making it suitable for resource-constrained settings such as CI pipelines.

1 Introduction

1.1 Problem Statement

When testing a polynomial function, or mechanism that retrieves one, it can be helpful to use random transforms on a data set to catch edge cases. However this introduces the probability that any given run of the test will fail to catch such an edge case.

The most common standard approach is to run the test a fixed number of times k, determined heuristically by the perceived risk of failure, and the cost of running the test.

However this approach has several drawbacks:

- Per-trial risk of failure is the quantity under measurement, thus rendering it difficult to determine a good k
- It does not provide a quantifiable confidence that the test has caught edge cases, or that the test has passed with a given level of confidence.
- It must use a conservative estimate for k, but cannot adapt to the observed failure rate during testing.
- k must be adjusted to compensate for changes in set-size, the effects of which can be difficult to estimate

It can be assumed a driving factor determining the rate of failure in such a setup is the presence of extreme outliers in the data set, and the interactions between these outliers. And in the case of artificial data sets, these outliers may be intentionally introduced to test the robustness of the mechanism under test, and thus have a known distribution and frequency.

Failure rate can in many cases be shown to scale with the size of the data set, but with diminishing returns as the data set grows larger.

While sequential analysis has a rich history in statistics [3], and Bayesian methods are well established, their application to software testing, particularly for numerical implementations under known injected noise, remains underexplored.

2 Proposed Solution

2.1 Bayesian Iterative Outlier Detection

We propose a modified Bayesian approach towards the probability of failure in any given data set, and a frequentist stopping criterion based on achieving a desired confidence level that sufficient iterations have been performed to catch edge cases caused by extreme outliers.

The algorithm takes the following parameters:

n	The size of the data set under test.
p	The probability of any individual point being an extreme outlier, chosen very conservatively.
q	The desired confidence that enough iterations have been performed to detect a theoretical bug caused by an extreme outlier.

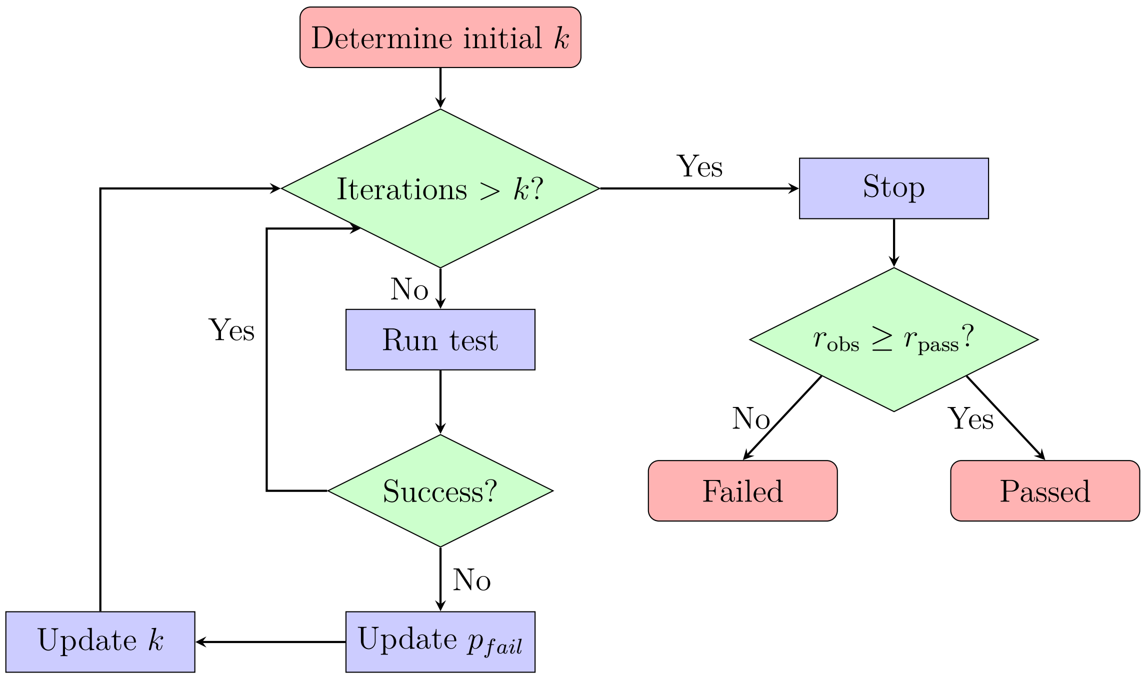


Figure 1: Flowchart of the BIOD testing procedure.

$r_{\textbf{pass}}$	The required ratio of data sets tested to pass for the overall procedure to be considered successful.

It also makes use of a pair of constants, used to tune the behaviour of the model. Derivations for suggested values are included below, in section 2.3:

$C = 1.864$	A damping constant reflecting the diminishing impact of additional data points on the likelihood of failure due to extreme outliers
$p_s = 4.43e5$	A strength parameter for the Bayesian prior, reflecting the confidence in the initial estimate of the failure probability

The user-defined risk parameters are the most critical to the success of the algorithm, as they directly impact the number of iterations required to achieve the desired confidence level. p should be chosen to be a very conservative underestimate of the true pointwise probability of extreme outliers, to ensure that the algorithm does not underestimate the required number of iterations. q should be chosen based on the user's tolerance for risk, with higher values indicating a lower tolerance for risk. Overestimating p can affect accuracy, while underestimation affects only worst-case compute.

For many users with standard use cases, q, the desired confidence level, and r_{pass}, the required pass ratio, could be conflated, since they both reflect differing aspects of the user's tolerance for risk. However they are kept separate to allow for more advanced use cases where the user may wish to set a high confidence level for catching edge cases, but a lower pass ratio to allow for known issues in the data set. The advantage of this is that the user can be more certain that any failures observed are due to real issues, rather than random chance.

The sections below describe the 3-phase approach for the BIOD method.

2.1.1 Estimation Phase

Given n the number of items per set, and p the pointwise lower-bound estimate for outlier frequency, we calculate an initial conservative estimate for the setwise probability of failure p_{fail}:

$$p' = \frac{p}{(1 + C \cdot n \cdot p)}, p_{\text{fail}} = 1 - (1 - p')^n \tag{1}$$

p' is used to model the saturation of the effect of individual outliers on setwise failure. It represents the inverse relationship $\frac{1}{np}$ between number of expected outliers, and the effects of interations between them. For more on the derivation of this form, and the recommended value for $C = 1.864$, see Section 2.3.1.

For small n the probability of failure is roughly linear in n, consistent with a binomial model where $p_{\text{fail}} \approx 1 - (1 - p)^n$. As n becomes large, the damping fraction saturates, and the effective failure probability approaches a limiting value $p_{\text{fail}} = 1 - e^{-1/C}$. In this regime, the number of failures per iteration can be approximated by a Poisson distribution with $\lambda = \frac{1}{C}$, reflecting the diminishing returns of increasing data set size on the likelihood of failure due to extreme outliers.

An initial estimate for the number of iterations k required to achieve the desired confidence q of having seen at least one failure can then be computed as:

$$k = \left\lceil \frac{\ln(1 - q)}{\ln(1 - p_{\text{fail}})} \right\rceil \tag{2}$$

Where p_{fail} is the probability of any single data set failing the test.

Finally, Bayesian priors for p_{fail} are calculated using p_s:

$$\alpha = p_{\text{fail}} \cdot p_s, \qquad \beta = (1 - p_{\text{fail}}) \cdot p_s \tag{3}$$

For more on the derivation of this form, and the recommended value for $p_s = 4.43e5$, see Section 2.3.2.

2.1.2 Iterative Testing Phase

In this phase we will track the following variables:

k The current estimate for iterations required to achieve the given confidence

α, β The priors for our Bayesian updates

iterations The number of iterations performed so far *passes* The number of successful iterations

passes_since_failure A tally of successes since the last failure

The test involves running a blackbox testing routine on a new set of size n in each iteration. We recommend each test utilize a seeded transform on an initial input set.

For each iteration, we first verify if $iterations \geq k$. If so, the test has ended and we move on to the output phase. We also check if a predetermined timeout has elapsed - if so, the testing process ends in failure, as there is insufficient evidence to support a claim.

Otherwise, we increment *iterations*, and perform the blackbox test on the next set; it should determine if a given set passes or fails.

If the set passes, we increment *passes* and $passes\_since\_failure$ and repeat the procedure. If the set fails, we update our estimates:

- Increment *alpha*, to represent the detected failure
- Perform $\beta + = passes\_since\_failure, passes\_since\_failure = 0$
- Recalculate the estimated setwise failure risk: $p_{\text{fail}} = \frac{\alpha}{\alpha+\beta}$
- Update the number of iterations required: $k = \left\lceil \frac{\ln(1-q)}{\ln(1-p_{\text{fail}})} \right\rceil$
- Repeat the iteration procedure

2.1.3 Output Phase

We can calculate the observed ratio of passes to failures r_obs as:

$$r_{\text{obs}} = \frac{\text{total passes}}{\text{total iterations}} \tag{4}$$

If $r_{\text{obs}} >= r_{\text{pass}}$, the test passes.

Due to the strong priors used in the Bayesian updates, the final beta distribution will be heavily influenced by the initial estimate of p_{fail}, and thus will not reflect the observed pass ratio. Thus, we can use a new beta distribution to calculate a credible interval for p_{fail} based on the observed data alone, only using the lower bound - since the upper bound estimate will have been biased by the sample size calculated using our conservative initial estimate.

$$p_{\text{fail,lower}} = F^{-1}\left(\frac{1-q}{2}; \text{total failures} + 1, \text{total passes} + 1\right) \tag{5}$$

$$\tag{6}$$

where F^{-1} is the Beta quantile function.

The outputs for the test are:

- The test result: pass / fail
- r_{obs}, the observed pass ratio
- The calculated lower bound on $p_{\text{fail, true}}$ given q

These can be communicated succinctly as: "[$100 \cdot r_{\text{obs}}$]% of tests passed. [$100 \cdot q$]% confidence that the true failure rate among datasets is at least $p_{\text{fail,lower}}$."

2.2 Limitations

The proposed method has several limitations and assumptions worth noting:

- The size of the data set n must be known in advance, and should remain constant throughout the testing process.
- The method assumes that the probability of extreme outliers is independent and identically distributed across the data set, which may not hold true in all cases.
- Very small n (< 100) require a very large number of iterations to converge; this is an expected biproduct of the model used, and we therefore recommend $n >= 100$

2.3 Derivations

2.3.1 The Damping Constant - C

As n increases, the naiive binomial form $p_{\text{fail}} = 1 - (1 - p')^n$ for expected failure trends to 100%. To resolve this, we introduce the damping fraction $p' = \frac{p}{(1 + C \cdot n \cdot p)}$

We began with the form

$$p' = \frac{p}{n\dot{p}} \tag{7}$$

In order to eliminate the undefined case at $n = 0$, and to ensure that $p' > p$ in all cases, we modified this to the form:

$$p' = \frac{p}{(1 + n \cdot p)} \tag{8}$$

And finally introduced the constant C to allow for tuning of the saturation effect:

$$p' = \frac{p}{(1 + C \cdot n \cdot p)} \tag{9}$$

C represents the binding strength modulating between outliers, and thus the rate of saturation of the effect of increasing n on p_{fail}.

The effect is the introduction of a asymptotic floor on the minimum number of iterations required for a set as $n \to \infty$, k_{floor}.

To find a reasonable value for C, we define the reasonable universe of parameters for the test space. For a theoretical set of infinite size, there is naturally a minimum number of iterations under which the test is not useful - k_{floor}. There is also a practical maximum number, above which the computation required for extremely large sets becomes infeasible.

We will define that practical range for k_{floor} to be between 3 and 10 iterations, and the we will define the lowest confidence q we consider reasonably conservative to be 0.8.

We we can analyze the effect of C effect on k_{floor} as n approaches infinity:

$$p' = \frac{p}{1 + Cnp} \tag{10}$$

$$p_{\text{fail}} = 1 - (1 - p')^n \tag{11}$$

For large n, $(1 - p')^n \to e^{-np'}$, so

$$p_{\text{fail}} = 1 - e^{-\frac{np}{1+Cnp}} = 1 - e^{-\frac{1}{C}\cdot\frac{Cnp}{1+Cnp}} \tag{12}$$

As $n \to \infty$, $\frac{Cnp}{1+Cnp} \to 1$, giving us $p_{\text{fail}} = 1 - e^{-1/C}$

Then

$$k = \frac{\ln(1-q)}{\ln(1-p_{\text{fail}})} \quad \Rightarrow \quad k_{\text{floor}} = \frac{\ln(1-q)}{\ln(e^{-1/C})} = \frac{\ln(1-q)}{-1/C} = -C\ln(1-q) \tag{13}$$

Solving for C:

$$C = \frac{k_{\text{floor}}}{-\ln(1-q)} \tag{14}$$

If we set $k_{\text{floor}} = 3$ as a reasonable minimum number of iterations that could be considered useful for a lower-bound on the value for q we consider reasonably conservative of $q = 0.8$, we find:

$$C = \frac{k_{\text{floor}}}{-\ln(1-q)} \quad \Rightarrow \quad C = \frac{3}{-\ln(1-0.8)} \approx 1.864$$

Additionally, since $k_{\text{floor}} = -C \cdot \ln(1-q)$, we can measure the effect of varying C on k_{floor} for a given q:

$$k_{\text{floor}} = -C\ \ln(1-q)$$

$$k_{\text{floor}} = -(1.864 - 0.5)\ \ln(1-0.8) \approx 2.19527$$
$$k_{\text{floor}} = -(1.864)\ \ln(1-0.8) \approx 2.99999$$
$$k_{\text{floor}} = -(1.864 + 0.5)\ \ln(1-0.8) \approx 3.80471$$

Taking into account the effect of ceil in the true calculation of k_{initial}, we can see that variations in C of approximately ± 0.5 around the nominal value of 1.864 produces a difference of approximately ± 1 in the final value, showing that this method is reasonably robust to small variations in C.

We will therefore recommend $C = 1.864$ as the nominal value for the damping constant.

To verify this value, we generated the k_{floor} produced by a variety of values within the defined parameters space, and then took C_{mean} and C_{σ} to produce the table and graph shown in Figure 5.

From the table we can see that the $C \approx 1.8$ produces values right in our defined practical range for k_{floor} across a variety of confidence levels, confirming our derived value.

If a usecase requires a different practical range for k_{floor} or q, a value for C should be selected using the formula above (13).

2.3.2 Prior Strength - ps

Isolating the effects of parameters on bias

Our first step to determining a reasonable p_s for a given model was to run the experiment described in Section 3.5, varying p_{fail} and r_{pass} over the base-case of $p_s = 1$, to determine if either parameter had a significant effect on bias.

These parameters were chosen since they are calculated directly from n and p, and thus would allow us to infer the effect of those parameters on bias.

For this and all experiments in this section we define bias as the difference between the test's observed pass ratio r_{obs}, and the true underlying pass rate for the simulated trial, r_{true}

$$\text{bias} = r_{\text{obs}} - r_{\text{true}} \tag{15}$$

A positive bias indicates the test overestimated the pass rate (false negative), while a negative bias indicates the test underestimated the pass rate (false positive).

Our hypothesis was that the delta between the expected level of failure p_fail and the acceptable level of failure $1 - r_{\text{pass}}$ would be a good predictor of bias.

However, as can be seen in Figure 6, neither parameter had a measurable effect on bias - which appeared to be randomly distributed around 0 witha constant error regardless of the values of p_{fail} and r_{pass}. Since p_{fail} encodes for k, and contains both n and p, we can infer that none of n, p, p_{fail}, or k have a significant effect on bias, which appears to be a set function for any inputs into the model given an unknown underlying failure rate.

Effect of prior strength on bias and compute

Our next step was to run a simplified experiment varying only p_s, shown in Figure 7a (a), to determine the effect of p_s on bias. This demonstrated the bias was indeed centered at 0, with a variance that decreased with increasing p_s, plateauing near $p_s = 1e7$.

We next ran the same experiment measuring the fraction of initially estimated iterations required to converge at various values of p_s, shown in Figure 7b (b). This demonstrated that increasing p_s also increased the required computation, again reaching an asymptotic limit, this time around $p_s = 1e8$.

Modeling the tradeoff between bias and compute

Following Experiment 3, we fit the relationship between the performed fraction of maximum iterations and p_s to a pair of Hill equations [2] of the form:

$$y = y_{min} + (y_{max} - y_{min})\frac{x^n}{k^n + x^n} \tag{16}$$

Where:

- y is the observed variable (bias stddev or compute fraction)
- x is p_s, the prior strength
- n is the Hill coefficient, which determines the steepness of the curve
- k is the value of x at which y reaches half of its maximum value
- y_{min} and y_{max} are the starting and ending values of y as x increases

Bias was first normalized to the maximum observed value, and compute was already a fraction of the maximum, making both variables suitable for fitting to this form.

The hill curve was chosen due to the need for a monotonic, saturating sigmoid function.

This resulted in a good fit for both bias and compute ($R^2 > 0.998$ for both curves).

The fits had the following parameters:

- Bias
  - $n = 0.548$
  - $k = 4.77 \cdot 10^4$
  - $y_{min} = 1.0$
  - $y_{max} = 0.323$
- Compute
  - $n = 0.639$
  - $k = 6.70 \cdot 10^5$
  - $y_{min} = 0.111$
  - $y_{max} = 1.0$

Selecting a recommended prior strength

We then defined a Lagrangian objective function [1] to balance the competing goals of minimizing bias and minimizing compute, introducing β, the tradeoff coefficient between the two objectives representing the relative importance of compute savings versus bias reduction:

$$\text{objective\_score}(p_s) = \text{bias\_std}(p_s) - \beta \cdot \text{compute\_frac}(p_s) \tag{17}$$

By sweeping β over logarithmic space from 10^{-5} to 10^{1}, or in other words, from a strong preference for minimizing bias to an equal weighting of both objectives, and solving for the root of the objective function, we can build a table of optimal p_s values for various tradeoff preferences.

The result, shown in Figure 9, demonstrates a flat region of optimal p_s values centered at $3.35 \cdot 10^{6}$, in which bias becomes relatively insensitive to p_s, while compute savings continue to improve significantly as p_s decreases. The tradeoff curve and suggested optimal value are shown in Figure 8.

We therefore recommend a nominal value of $p_s = 3.35 \cdot 10^{6} \pm 1.98 \cdot 10^{4}$ for a good balance between compute and bias. Compared to the largest reasonable value of $p_s = 1e7$, this value results in only a 3% increase in bias stddev, while saving 24% of worst-case compute. By comparison, the less conservative value of $p_s = 4.43e5$ results in a 12% increase in bias stddev, while saving 39% of worst-case compute.

The recommendation corresponds to $\beta = 0.51$, meaning that compute savings are considered roughly half as important as bias reduction.

It is noteworthy that since compute is a fraction of the initially estimated iterations actually completed, this will already scale with n; The larger the set size, the greater the potential savings as compared to the initially estimated worst-case compute. Thus, this recommended value for p_s should be robust across a wide range of n values.

3 Methodology

3.1 Experiment 1: Conservative Estimate of p_fail vs Simulated True p_fail

Objective. To verify that calculated p_{fail} is a conservative underestimate of the true simulated p_{fail} across a range of dataset sizes n.

3.1.1 Setup.

- Vary set size n logarithmically from $1e0$ to $1e5$.
- For each n, run 10,000 trials.

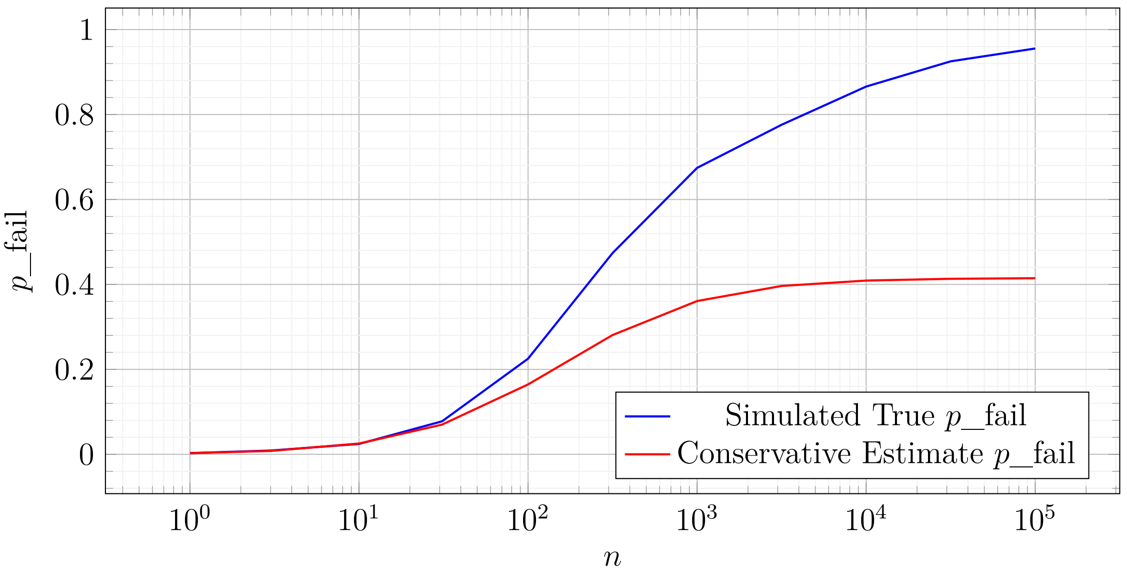


Figure 2: Graph showing the conservative estimates of p_{fail} vs simulated true p_{fail} for various dataset sizes n

- For each trial, simulate failure due to extreme outliers, accounting for interactions between outliers:
  - Draw n samples from a standard normal distribution ($\mu = 0$, $\sigma = 1$).
  - Define extreme outliers as samples with absolute value greater than 3.
  - A trial fails if the sum of all extreme outliers exceeds 3 in absolute value.
- Record the observed failure rate across trials as the simulated true p_{fail}.
- For each n, calculate the corresponding conservative initial estimate of p_{fail} using the BIOD method:
  - Confidence level: 0.99
  - Outlier probability: 0.0027 (approximate probability of $|x| > 3\sigma$ in a normal distribution)
  - Pass ratio: 0.0 (not used in this experiment)

3.1.2 Definition of Metrics.

- $p_{\text{fail, true}}$: Observed failure rate from simulations.
- $p_{\text{fail, calc}}$: Calculated conservative estimate of p_{fail} using BIOD.

3.1.3 Hypothesis.

BIOD is expected to yield a calculated $p_{\text{fail, calc}}$ that is less than or equal to the observed $p_{\text{fail, true}}$ across all tested set sizes n.

3.1.4 Results.

As shown in Figure 2, $p_{\text{fail, calc}}$ consistently underestimates $p_{\text{fail, true}}$ across the full range of dataset sizes n. Moreover, the two curves exhibit similar functional form, indicating that the estimator preserves the underlying dependence on n while remaining systematically conservative.

3.1.5 Analysis / Interpretation.

The results confirm the hypothesis that the BIOD-calculated p_{fail} is a conservative underestimate of the true simulated p_{fail} across all dataset sizes tested.

By following the same trend as the true p_{fail}, the BIOD estimate avoids the trap of underestimating by enough to make initial k too large to be useful, as n increases, while still providing a reliable conservative estimate.

3.1.6 Summary / Recommendation.

The BIOD method provides a reliable conservative estimate of p_{fail}, without trend inversion as n varies, making it suitable for this application.

3.2 Experiment 2: Predicted k to Observed k to Failure

Objective. To verify that the predicted number of iterations k to failure from BIOD encompasses the observed number of iterations k to failure across a range of dataset sizes n.

3.2.1 Setup.

- Vary set size n logarithmically from $1e0$ to $1e5$.
- For each n, run 10,000 trials.
  - Draw n samples from a standard normal distribution ($\mu = 0$, $\sigma = 1$).
  - Define extreme outliers as samples with absolute value greater than 3.
  - A trial fails if the sum of all extreme outliers exceeds 3 in absolute value.
- For each trial, record the number of iterations k taken to find a failing set

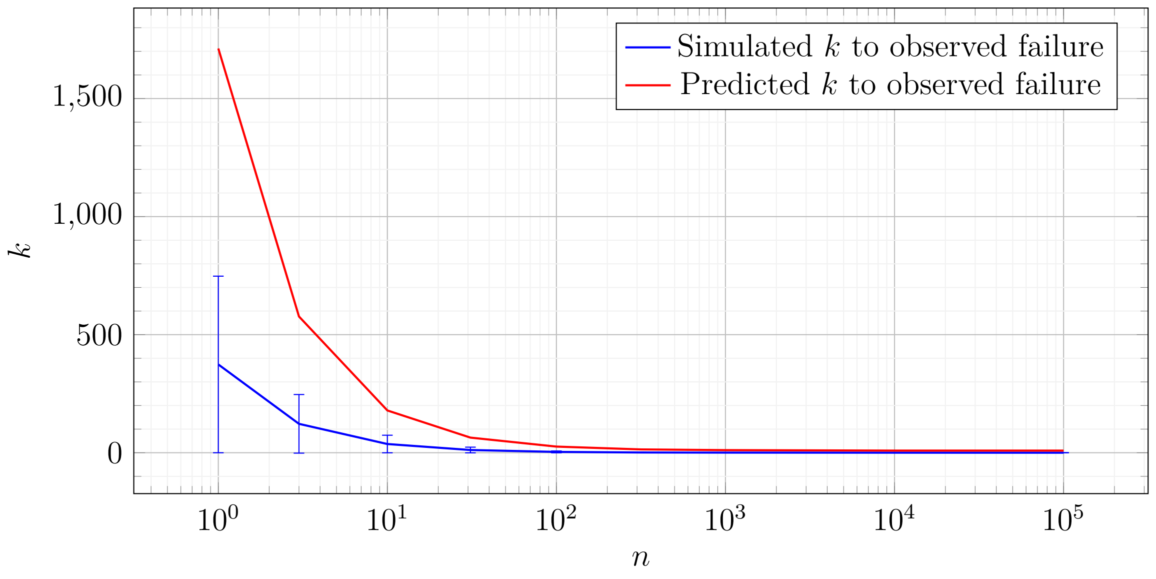


Figure 3: Graph showing the predicted number of iterations k to observed failure vs the simulated results

- Take the mean and standard deviation of observed k across trials.
- After all trials for a given n, calculate the predicted k to failure using the BIOD method:
  - Confidence level: 0.99
  - Outlier probability: 0.0027 (approximate probability of $|x| > 3\sigma$ in a normal distribution)
  - Pass ratio: 0.0 (not used in this experiment)

3.2.2 Definition of Metrics.

- $k_{\text{obs, mean}}$: Mean observed iterations to failure from simulations.
- $k_{\text{obs, stdev}}$: Standard deviation of observed iterations to failure from simulations.
- $k_{\text{predicted}}$: Predicted iterations to failure using BIOD.

3.2.3 Hypothesis.

The BIOD is expected to yield a predicted $k_{\text{predicted}}$ that is greater than or equal to the mean observed $k_{\text{obs, mean}}$ across all tested set sizes n.

3.2.4 Results.

As shown in Figure 3, $k_{\text{predicted}}$ consistently overestimates $k_{\text{obs, mean}}$ across the full range of dataset sizes n. The predicted values remain above the observed means, following the same trend as observed k to failure, similarly to Experiment 1.

The observed standard deviation $k_{\text{obs, stdev}}$ shrinks as n increases, indicating more consistent behavior in larger datasets.

3.2.5 Analysis / Interpretation.

The results confirm the hypothesis that the BIOD-predicted k to failure is a conservative overestimate of the mean observed k to failure across all dataset sizes tested.

This means that for a well specified test, BIOD's predicted k will likely include a failure within that number of iterations, providing confidence in the method's reliability.

3.2.6 Summary / Recommendation.

The BIOD method provides a reliable conservative prediction of iterations to failure, making it suitable for this application.

3.3 Experiment 3: Confidence Interval on p__fail Estimates

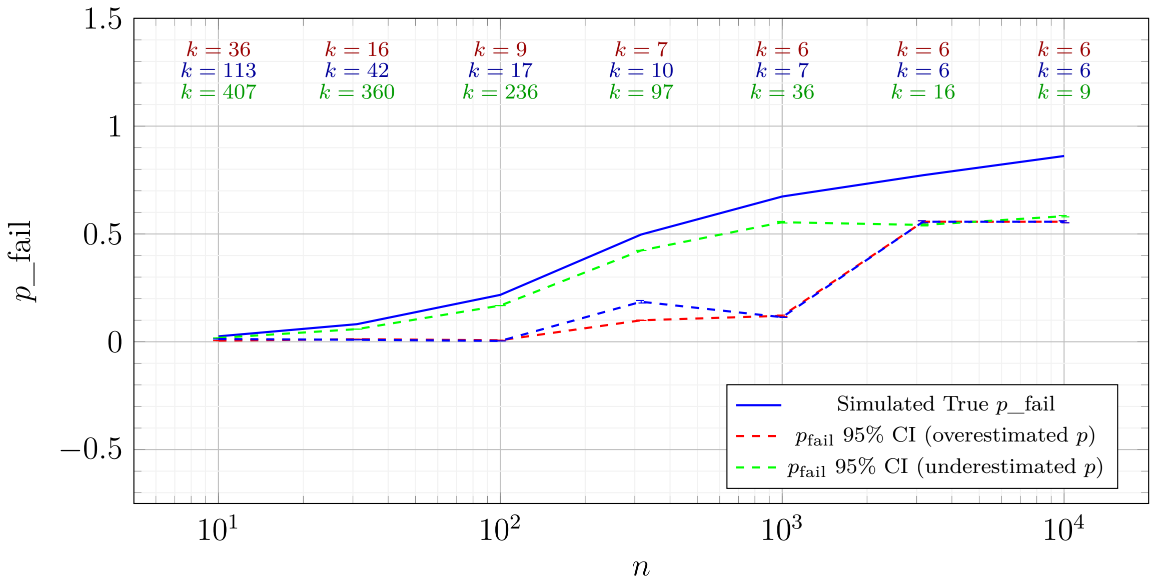


Figure 4: Graph showing the mean and standard deviation of lower-bound confidence intervals on the sample of iterations observed by BIOD from a simulation of 10,000 trials, compared to the simulated true p_{fail} for various dataset sizes n, and for various values of pointwise p. Green was deliberately underestimated, blue was exact, and red was deliberately overestimated. Markers indicate the number of iterations k performed by BIOD at each point.

Objective. To verify that the calculated p_{fail} from BIOD provides accurate confidence intervals that encompass the observed p_{fail} across a range of dataset sizes n and pointwise p values.

3.3.1 Setup.

- Vary set size n logarithmically from $1e0$ to $1e5$.
- Test 10,000 sets for each combination of n, to calculate observed p_{fail}.
- Run BIOD with the same results to calculate confidence intervals on p_{fail} for each value of pointwise p:
  - Underestimate of p: $p_{\text{outlier}} = 1e - 4$
  - Exact p: $p_{\text{outlier}} = 0.0027$ (approximate probability of $|x| > 3\sigma$ in a normal distribution)
  - Overestimate of p: $p_{\text{outlier}} = 0.01$

3.3.2 Definition of Metrics.

- p_{fail}: True failure rate from simulations.
- $p_{\text{fail, biod, min}}$: Lower bound of BIOD-calculated confidence interval for p_{fail}.
- $p_{\text{fail, biod, max}}$: Upper bound of BIOD-calculated confidence interval for p_{fail}.
- $p_{\text{fail, obs}}$: Observed failure rate in the sample BIOD selects.

3.3.3 Hypothesis.

The BIOD-calculated confidence intervals for p_{fail} are expected to encompass the observed p_{fail} for accurate and underestimated pointwise p values, and to potentially miss for overestimated pointwise p values, across all tested set sizes n.

3.3.4 Results.

As shown in Figure 4, the calculated confidence intervals for p_{fail} from BIOD successfully encompass the observed p_{fail} for both all tested set sizes n even for badly specified pointwise p values.

When pointwise p is underestimated (green), the confidence intervals are narrower and more accurate, closely tracking the observed p_{fail}.

When pointwise p is exact (blue), the confidence intervals remain accurate, but is much wider than the underestimated case.

When pointwise p is overestimated (red), the confidence intervals are significantly wider for small n and converge to the exact case as n increases, but still encompass the observed p_{fail}.

3.3.5 Analysis / Interpretation.

These results show that BIOD is robust to badly specified pointwise p parameters, but gains significant accuracy when pointwise p is well specified or conservative (i.e., underestimated).

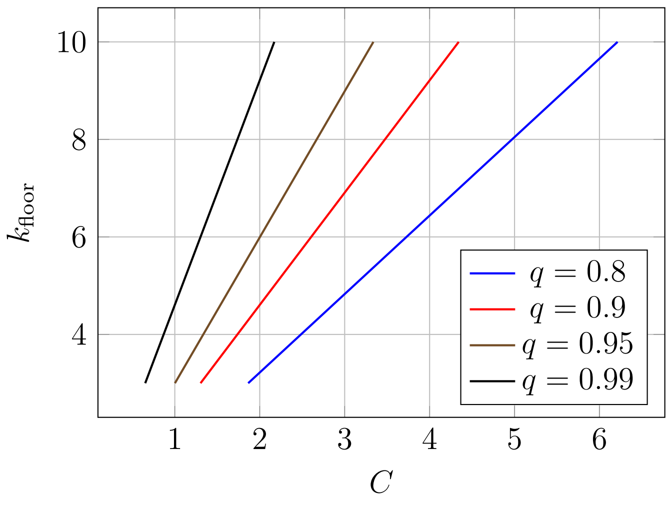
An overly conservative (underestimated) pointwise p leads to tighter confidence intervals, due to the larger number of iterations BIOD performs in this case, improving the accuracy of the p_{fail} estimate.

An overly aggressive (overestimated) pointwise p leads to wider confidence intervals, especially for small n, due to the smaller number of iterations BIOD performs in this case, reducing the accuracy of the p_{fail} estimate. However, as n increases, the effect of pointwise p mis-specification diminishes, and the confidence intervals converge towards those obtained with exact pointwise p.

3.3.6 Summary / Recommendation.

The BIOD method provides reliable confidence intervals for p_{fail} estimates, particularly when pointwise p is well specified or underestimated, but is still useful when pointwise p is overestimated.

3.4 Experiment 4: Damping Constant Effect on k floor



$C\backslash q$	0.8	0.9	0.95	0.99
1.56	3	4	5	8
1.82	3	5	6	9
2.07	4	5	7	10
2.33	4	6	7	11
2.59	5	6	8	12
2.84	5	7	9	14
3.1	5	8	10	15
3.36	6	8	11	16
3.61	6	9	11	17
3.87	7	9	12	18
4.13	7	10	13	20

Figure 5: Left: Graph of k_{floor} vs C values in the defined parameter space. Right: Table showing values of k_{floor} for $C_{\text{mean}} \pm C_{\sigma}$ for various confidence levels q.

Objective. To investigate the relationship between the damping constant C, desired k floor and confidence level q.

3.4.1 Setup.

The experiment will be performed in 2 parts:

Part 1 Vary q smoothly from 0.8 to 0.99 in increments of 0.01. For each q, vary k from 3 to 10 in increments of 1. Compute the corresponding C value for each (k, q) pair using the formula provided below:

$$C = \frac{k}{-\ln(1-q)}$$

This will provide a range of C values corresponding to the desired k floor and q values defined in the parameter space.

Part 2 Use the results from Part 1 to compute the mean C_{mean} and standard deviation C_σ of the damping constant C across all (k, q) pairs. For the range $C = C_{\text{mean}} \pm C_\sigma$, compute the corresponding k_{floor} values for specific confidence levels $q = [0.8, 0.9, 0.95, 0.99]$:

$$k_{\text{floor}} = C \cdot -\ln(1-q)$$

This will provide a table of k_{floor} values for the defined range of C and specific confidence levels.

3.4.2 Definition of Metrics.

Damping constant C for each desired k floor and pass probability q.

3.4.3 Hypothesis.

We expect C to increase monotonically with k and q.

3.4.4 Results.

The results, shown in Figure 5, confirm that C increases with both desired k floor and pass probability q, ranging from < 1 to > 6 in our defined parameter space.

Narrowing the range to $C = C_{\text{mean}} \pm C_\sigma$ gives us a range of 1.56 to 4.13, where k_{floor} values between 3 and 10 correspond to C values between 1.82 and 2.07 for confidence levels between 0.8 and 0.99.

3.4.5 Analysis / Interpretation.

The results confirm the hypothesis that C increases with both desired k floor and pass probability, and gives us a practical basis to verify the value found analytically in Section 2.3.1.

3.4.6 Summary / Recommendation.

From the table, a value between $C = 1.82$ and $C = 2.07$ best represents the desired k floor range of 3-10 for confidence levels between 0.8 and 0.99.

3.5 Experiment 5: Bias vs p_fail and ratio_pass

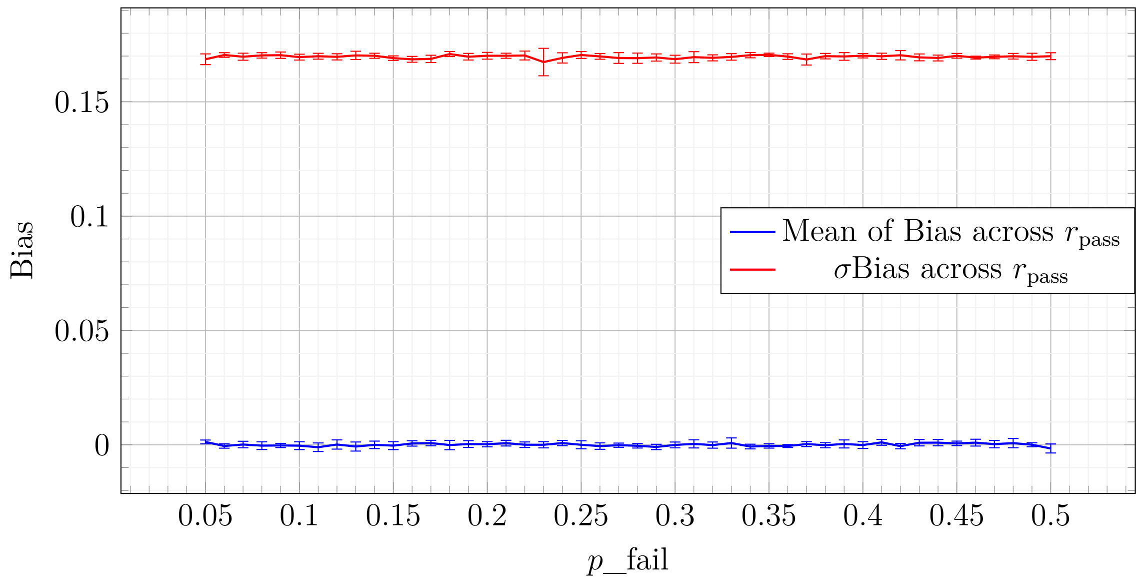


Figure 6: Graph demonstrating the lack of correlation between p_fail, pass$_\text{r}$atio, and bias in the base case of $p_s = 1$

Objective. To investigate the relationship between bias, p_{fail}, and pass ratio. p_{fail} is calculated using p, q and n, and so betwen it and pass$_\text{r}$atio we can explore the bias behavior of BIOD across the parameter space in the base-case of $p_s = 1$.

3.5.1 Setup.

- We select pass$_\text{r}$atio $\in \{0.2, 0.5, 0.8, 0.9, 0.95, 0.99, 0.999\}$.
- We vary p_{fail} from 0.05 to 0.50 in increments of 0.01

- For each p_{fail}, we compute the corresponding dataset size n using:

$$n = \left\lceil \frac{-\ln(1 - p_{\text{fail}})}{p'} \right\rceil, \quad p' = \frac{1}{1 + C \cdot n \cdot p}, \quad C = 1.864, \quad p = 10^{-6}$$

For each $(p_{\text{fail}}, \textit{ratio\_pass})$ pair, perform the 10,000 iterations of the following:

- Draw a value x from a uniform distribution $U(0, 1)$.
- Determine range for the rate of failure based on the following piecewise function:

$$\text{failure}_{\text{r}}\text{ate} = \begin{cases} [0.001, 0.01] & x < 0.1 \\ [0.01, 0.05] & 0.1 \leq x < 0.3 \\ [0.05, 0.20] & 0.3 \leq x < 0.65 \\ [0.20, 0.50] & 0.65 \leq x < 0.9 \\ [0.50, 0.999] & x \geq 0.9 \end{cases}$$

- Draw a true failure probability $p_{\text{fail, true}}$ from a uniform distribution over the determined range.
- Run BIOD using the failure rate found, noting the pass$_{\text{r}}$atio value at the end of the run, the initial estimate for k and the number of iterations actually performed.
- Compute bias as $p_{\text{fail, true}} - \text{pass}_{\text{r}}\text{atio}$.
- Compute the mean and standard deviation of bias across all iterations for each pass$_{\text{r}}$atio value.

For each p_{fail}, compute:

- The mean of all bias means across all pass$_{\text{r}}$atio values.
- The standard deviation of all bias means across all pass$_{\text{r}}$atio values.
- The mean of all bias standard deviations across all pass$_{\text{r}}$atio values.
- The standard deviation of all bias standard deviations across all pass$_{\text{r}}$atio values.

3.5.2 Definition of Metrics.

- p_{fail}: Calculated failure probability from BIOD.
- pass$_{\text{r}}$atio: Pass ratio parameter provided to BIOD.
- *bias*: Difference between true failure probability and observed pass ratio. The mean of bias represents systematic error, while the standard deviation represents variability in the error.

3.5.3 Hypothesis.

We expect bias to increase proportionally with the difference between p_{fail} and pass_ratio.

3.5.4 Results.

Contrary to our hypothesis, we find that bias remains relatively constant across the parameter space, with mean bias values clustering around 0.0 and standard deviation of bias remaining low for all combinations of p_{fail} and pass_ratio.

We also found that the standard deviation of bias mean and standard deviation of bias standard deviation across all pass_ratio values remained low for all p_{fail} values.

3.5.5 Analysis / Interpretation.

Since the parameters tested involve all of BIOD's inputs except for prior strength, this experiment establishes that BIOD's bias behavior is not predictable from inputs, and will be a factor of prior strength alone.

3.5.6 Summary / Recommendation.

These results indicate that prior strength is the key parameter influencing bias in BIOD, and should be the focus of further investigation.

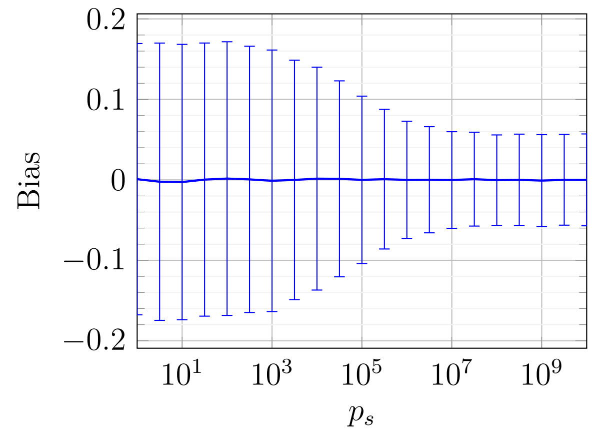
It suggests that p_s is not a function of inputs, but a constant that can be pre-computed to achieve desired bias characteristics.

3.6 Experiment 6: Prior Strength vs Bias and Compute

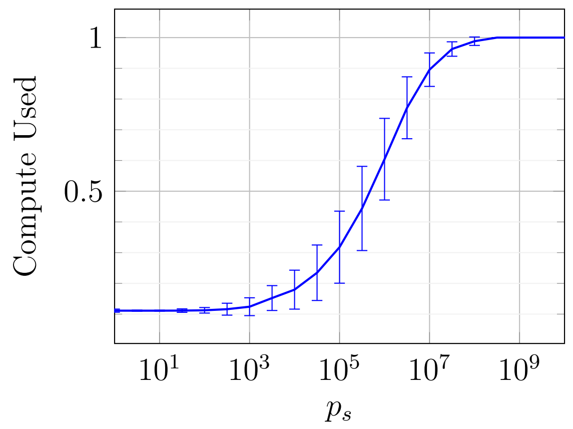
Objective. To investigate the relationship between prior strength, bias, and computational cost in BIOD.

3.6.1 Setup.

- We vary prior strength p_s from $10e0$ to $10e12$ in logarithmic increments of $10e0.5$.
- For each p_s, we perform 10,000 iterations of the following:
  - Draw a value x from a uniform distribution $U(0, 1)$.



(a) Bias vs increasing p_s



(b) $\frac{iterations}{k_{initial}}$ vs p_s

Figure 7: Comparison of bias behavior and convergence across p_s values

- Determine range for the rate of failure based on the following piecewise function:

$$\text{failure}_\text{rate} = \begin{cases} [0.001, 0.01] & x < 0.1 \\ [0.01, 0.05] & 0.1 \leq x < 0.3 \\ [0.05, 0.20] & 0.3 \leq x < 0.65 \\ [0.20, 0.50] & 0.65 \leq x < 0.9 \\ [0.50, 0.999] & x \geq 0.9 \end{cases}$$

- Draw a true failure probability $p_{\text{fail, true}}$ from a uniform distribution over the determined range.
- Run BIOD using the failure rate found, noting the $\text{pass}_\text{r}\text{atio}$ value at the end of the run, the initial estimate for k and the number of iterations actually performed.
- Compute bias as $p_{\text{fail, true}} - \text{pass}_\text{r}\text{atio}$.
- Compute the fraction of initial k actually performed: $\frac{k_{\text{performed}}}{k_{\text{initial}}}$.

• Compute the mean and standard deviation of bias across all iterations.

• Compute the mean and standard deviation of fraction of initial k performed across all iterations.

3.6.2 Definition of Metrics.

• p_s: Prior strength parameter provided to BIOD.

• *bias*: Difference between true failure probability and observed pass ratio. The mean of bias represents systematic error, while the standard deviation represents variability in the error.

• *compute_fraction*: Fraction of initial k actually performed, representing computational cost.

3.6.3 Hypothesis.

We expect bias to remain centered around 0.0, but its standard deviation to decrease as prior strength p_s increases. We also expect computational cost to increase with prior strength, as higher p_s values lead to larger initial k estimates.

3.6.4 Results.

As shown in Figure 7b, the mean bias remains centered around 0.0 across all prior strength values tested, confirming part of our hypothesis.

However, while the standard deviation of bias does decrease with increasing prior strength, there is a ceiling effect observed around $p_s = 10e7$ beyond which further increases in prior strength yielded no additional reduction in bias variability, while still increasing computational cost.

3.6.5 Analysis / Interpretation.

There appears to be a sigmoidal saturation effect in bias standard deviation with respect to both bias standard deviation and fraction of worst-case compute performed.

This suggests there exists an optimal prior strength value that balances bias reduction and computational cost, beyond which further increases in prior strength yield diminishing returns in bias improvement while incurring higher computational costs.

3.6.6 Summary / Recommendation.

Further analysis is needed to precisely identify the optimal prior strength value that minimizes bias variability while controlling computational cost.

3.7 Experiment 7: Prior Strength Derivation

Objective. To use the results from the experiment in Section 3.6 to derive a recommended prior strength value for BIOD based on desired bias characteristics. Based on the monotonic, sigmoidal behavior observed, we selected the hill equation as a suitable model.

3.7.1 Setup.

For the curve fits below, we will use the following hill curve function [2]:

$$y = y_{min} + (y_{max} - y_{min}) \frac{x^n}{k^n + x^n}$$

where:

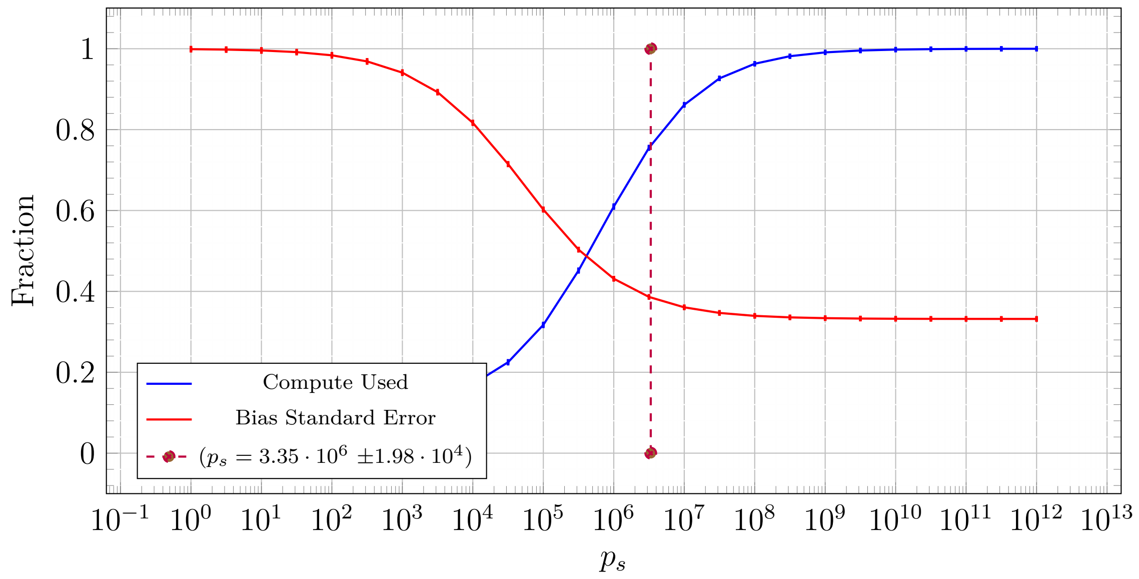


Figure 8: The compute / accuracy tradeoff in selecting p_s. Compute used is in the fraction of $k_{initial}$ actually performed in simulation, Bias is in the standard deviation of bias observed. Markers indicate recommended values for p_s.

Balance Factor	p_s	Compute	Bias	Δ Compute	Δ Bias	p_s Error
0.44	$7.08 \cdot 10^6$	0.83	$3.49 \cdot 10^{-2}$			$3.83 \cdot 10^4$
0.47	$4.71 \cdot 10^6$	0.8	$4.38 \cdot 10^{-2}$	$3.72 \cdot 10^{-2}$	$8.85 \cdot 10^{-3}$	$2.70 \cdot 10^4$
0.51	$3.35 \cdot 10^6$	0.76	$5.28 \cdot 10^{-2}$	$3.56 \cdot 10^{-2}$	$9.02 \cdot 10^{-3}$	$1.98 \cdot 10^4$
0.54	$2.48 \cdot 10^6$	0.73	$6.2 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$9.22 \cdot 10^{-3}$	$1.46 \cdot 10^4$
0.58	$1.9 \cdot 10^6$	0.69	$7.14 \cdot 10^{-2}$	$3.25 \cdot 10^{-2}$	$9.42 \cdot 10^{-3}$	$1.64 \cdot 10^4$
0.62	$1.49 \cdot 10^6$	0.66	$8.11 \cdot 10^{-2}$	$3.11 \cdot 10^{-2}$	$9.64 \cdot 10^{-3}$	$1.25 \cdot 10^4$
0.67	$1.19 \cdot 10^6$	0.63	$9.09 \cdot 10^{-2}$	$2.97 \cdot 10^{-2}$	$9.86 \cdot 10^{-3}$	$9.80 \cdot 10^3$
0.71	$9.7 \cdot 10^5$	0.61	0.1	$2.84 \cdot 10^{-2}$	$1.01 \cdot 10^{-2}$	$7.80 \cdot 10^3$
0.77	$7.97 \cdot 10^5$	0.58	0.11	$2.71 \cdot 10^{-2}$	$1.03 \cdot 10^{-2}$	$6.31 \cdot 10^3$
0.82	$6.62 \cdot 10^5$	0.55	0.12	$2.59 \cdot 10^{-2}$	$1.06 \cdot 10^{-2}$	$5.19 \cdot 10^3$

Figure 9: Table of optimal prior strength values for various balance factors β weighing bias reduction vs computational cost, based on hill curve fits to the normalized standard deviation of bias and normalized fraction of worst-case compute performed data.

- y: Normalized value at a given p_s.
- x: Prior strength p_s.
- y_{min}: Normalized value at $p_s = 1$.
- y_{max}: Normalized value at $p_s = 10^{12}$.
- k: Half-maximal effective concentration, representing the p_s value at which saturation is at 50%.

- n: Hill coefficient, describing the steepness of the curve.

- Normalize the standard deviation of bias results to be a fraction of the maximum observed standard deviation across all prior strength values tested.
- Fit the hill curve function above to the normalized standard deviation of bias data using non-linear least squares optimization to find the best-fit.
- Fit the hill curve function above to the fraction of worst-case compute performed data using non-linear least squares optimization to find the best-fit.
- Calculate the r^2 value for each fit to assess goodness of fit.
- Introduce a balance factor β and a Lagrangian objective function [1] to weigh the importance of bias reduction vs computational cost:

$$\text{objective}_\text{s}\text{core}(p_s) = \text{bias}_\text{s}\text{td}(p_s) - \beta \cdot \text{compute}_\text{f}\text{rac}(p_s)$$

- For a range of 200 β values in the logarithmic space from 10^{-5} to 10^1, find the prior strength p_s that minimizes the objective function.
- Only consider solutions where p_s is in the observed range of 10^0 to 10^{12}.
- For each optimal p_s found, find the error on p_s using the delta method:

$$\text{error}(p_s) = \frac{\sigma hill^2_\text{bias} + (\beta \cdot \sigma hill^2_\text{compute})}{\frac{\partial\, \text{objective}(p_s, \beta)}{\partial p_s}}$$

- Build a table of optimal p_s values for each β with the following columns:
  - (1) β: Balance factor.
  - (2) p_s: Optimal prior strength for the given β.
  - (3) Normalized fraction of worst-case compute performed at optimal p_s.
  - (4) Normalized standard deviation of bias at optimal p_s.
  - (5) Delta between current and previous row's normalized fraction of worst-case compute performed.
  - (6) Delta between current and previous row's normalized standard deviation of bias.
  - (7) The error on p_s

3.7.2 Definition of Metrics.

- p_s: Prior strength parameter provided to BIOD.
- Normalized standard deviation of bias: Standard deviation of bias normalized to the maximum observed standard deviation across all prior strength values tested.
- Normalized fraction of worst-case compute performed: Fraction of worst-case compute performed normalized to the maximum observed fraction across all prior strength values tested.
- r^2: Coefficient of determination for the hill curve fits, indicating goodness of fit.
- β: Balance factor weighing bias reduction vs computational cost.
- n, k: Hill curve parameters as defined above.
- $p_{\mathrm{s},optimal}$ Recommendation prior strength for a given β.
- error($p_{\mathrm{s},optimal}$): Estimated error on the recommended prior strength.

3.7.3 Hypothesis.

Based on the monotonic, sigmoidal behavior and apparent saturation effects observed in Section 3.6, we expect the hill equation to provide a good fit to both the normalized standard deviation of bias and normalized fraction of worst-case compute performed data, with fair r^2 values indicating strong goodness of fit.

We also expect to find a fairly wide range of optimal prior strength p_s values corresponding to a flat region in bias, with the error on p_s estimates being moderate, contributing to a wider recommendation range.

3.7.4 Results.

The hill curve fits yielded high r^2 values:

- Normalized standard deviation of bias fit: $r^2 = 0.9989$
- Normalized fraction of worst-case compute performed fit: $r^2 = 0.9991$

The table of optimal prior strength p_s values for various balance factors β revealed a range of p_s values corresponding to a flat region in bias where further increases in prior strength yielded minimal bias improvement while incurring higher computational costs.

It also revealed small errors (most $< 0.5\%$) on the values of both fits, and by extension, on the recommended prior strength values.

The table shown in Figure 9 shows a value of $p_s = 3.35 \cdot 10^6 \pm 1.98 \cdot 10^4$ provides a good balance between bias reduction and computational cost, yielding a normalized standard deviation of bias of 0.052 and a normalized fraction of worst-case compute performed of 0.76.

3.7.5 Analysis / Interpretation.

Both r^2 exceeding 0.998 confirm the hill equation is an excellent model for both the normalized standard deviation of bias and normalized fraction of worst-case compute performed data. The tight errors on the fits further reinforce the reliability of the model.

The table of optimal prior strength p_s values indicates a practical range for selecting p_s where the change in bias reduction becomes negligible compared to the increase in computational cost, allowing users to make informed decisions based on their specific bias tolerance and compute resource constraints.

The recommended value provides an option where increasing p_s to the maximum reasonable value yields 5% improvement in bias standard deviation at the cost of 24% increase in computational cost.

The $\beta = 0.51$ finding at the suggested p_s corresponds to a value determining the relative importance of bias reduction vs computational cost, that compute here is roughly half as important as bias reduction.

3.7.6 Summary / Recommendation.

We recommend a prior strength value of $p_s = 3.35 \cdot 10^6 \pm 1.98 \cdot 10^4$ for BIOD, balancing bias reduction and computational cost effectively for most applications.

The small errors on both the curve fits and recommended prior strength values provide confidence in the reliability of this recommendation.

4 Results

We evaluated the BIOD algorithm across a range of scenarios to assess its performance in terms of confidence accuracy, computational efficiency, and robustness to parameter mis-specification.

In the experiments described in Sections 3.1 and 3.2, we explored the effectiveness of BIOD's initial failure probability estimation compared to simulated ground truth values. Figures 2 and 3 illustrate that BIOD's estimates closely align with the actual failure probabilities, while remaining conservative, thus validating the algorithm's reliability in practical applications.

In the subsequent experiment in Section 3.3, we tested the validity of the confidence interval BIOD uses to determine when to stop iterating. As shown in Figure 4, the lower-bound confidence intervals produced by BIOD consistently encompass the true failure probabilities across various dataset sizes and pointwise failure probabilities, demonstrating the robustness of the stopping criterion.

In addition, contrary to our hypothesis, the badly specified pointwise p values (blue and red curves) performed comparably to the well-specified values (green curves), indicating that BIOD is resilient to inaccuracies in initial parameter settings.

Next, in Section 3.4, we explored the effect of the damping constant on BIOD's stopping condition via the floor on the number of iterations as set size increases. Figure 5 shows that the damping constant has a linear monotonic effect on the number of iterations performed, allowing for tuning based on available computational resources.

Finally, in Sections 3.5 through 3.7, we investigated the impact of prior strength on BIOD's performance. Figure 8 shows that increasing prior strength leads to an improvement in bias of the failure probability estimates, albeit with diminishing returns, at the cost of increased computational effort following the hill curve for both metrics.

These results collectively demonstrate that BIOD is an effective and adaptable algorithm for iterative outlier detection, capable of providing reliable estimates while allowing for customization based on specific application needs.

5 Discussion

The results presented in Section 4 highlight several key strengths of the BIOD algorithm, as well as areas for further exploration.

One notable finding is BIOD's robustness to mis-specification of the pointwise failure probabilities. Contrary to our initial hypothesis, the performance of BIOD remained consistent regardless of whether the pointwise p values were well-specified or deliberately misestimated. This resilience suggests that BIOD can be effectively applied in scenarios where precise prior knowledge of failure probabilities is unavailable, enhancing its practical utility in a wider range of applications than initially anticipated.

Two possible explanations for this robustness are the conservative nature of the initial failure probability estimates due to the damping fraction model, as well as the effect of the relatively strong prior strength used in the Bayesian updating process (Section 3.7). Both factors likely contribute to stabilizing the estimates against inaccuracies in the initial parameters.

Another finding was the lack of inherent bias in BIOD's failure probability estimates across various scenarios (See Section 3.5). Across the universe of input parameters tested, BIOD consistently produced a mean bias near zero, with very little variation. This characteristic is particularly advantageous in applications where unbiased estimates are critical, as it ensures that the algorithm does not systematically overestimate or underestimate failure probabilities regardless of specified parameters.

The introduction of the damping fraction to temper the number of iterations as set size increases also proved effective in controlling computational cost while maintaining reliable estimates. The model represents a saturating monotonic relationship with C modulating the effective binding strength between outliers in the dataset. This allows users to tune the algorithm based on available computational resources without significantly compromising accuracy. It is noteworthy that p_s and C should be considered a pair - modifying C will likely change the optimal p_s value.

During the update procedure, instead of a pure Bayesian update after each iteration, we employed a bulk update approach, aggregating evidence from multiple iterations before updating the posterior distribution on failure. This has the advantage of eliminating the possibility of artificially lengthy tests in the case of an absense of failure without compromising the Bayesian nature of the update.

Finally, the use of a frequentist stopping criterion based on confidence intervals proved effective in ensuring that BIOD halts iterations once sufficient evidence has been gathered. The lower-bound confidence intervals consistently encompassed the true failure probabilities, demonstrating the reliability of this stopping condition. This approach balances the need for computational efficiency with the requirement for accurate estimates, making it a practical choice for real-world applications.

Future work could explore the extend of the independence assumption made in the BIOD model, particularly in datasets where outliers may exhibit correlations. Additionally, investigating alternative prior distributions or adaptive prior strength mechanisms could further enhance the algorithm's performance across diverse scenarios.

6 Conclusion

We have presented the Bayesian Iterative Outlier Detection (BIOD) algorithm, a method for adaptively determining the number of iterations required to achieve a desired confidence level in randomized testing. By combining Bayesian statistics with iterative refinement based on observed failure rates, BIOD improves upon traditional fixed-iteration approaches.

Empirical evaluation shows that BIOD is robust to poorly specified parameters and accurately models failure rates under independent, normally distributed errors. It balances computational cost with confidence in results, making it practical for resource-constrained settings such as CI pipelines. Future work could explore extending BIOD to correlated or non-normal error models, further broadening its applicability.

References

[1] Hugh Everett. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Oper. Res.*, 11(3):399–417, June 1963.

[2] Rudolf Gesztelyi, Judit Zsuga, Balázs Varga, Bela Juhasz, and Arpad Tosaki. The hill equation and the origin of quantitative pharmacology. *Archive for History of Exact Sciences*, 66, 07 2012.

[3] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

Appendix

Code

Rust Implementation

The repository for this paper contains a Rust implementation of the BIOD algorithm used for the experiments described. The code can be found at: `https://github.com/rscarson/baysian_iterative_outlier_detection/`

It is a ready-to-use implementation in the form of a crate named `biod`, which can be included in other Rust projects via Cargo.

Python Implementation

This code and a sample run can be found at: `https://github.com/rscarson/baysian_iterative_outlier_detection/blob/master/biod.py`

```python
import math
import random
import sys
import time

from scipy.stats import beta

DAMPING_CONSTANT = 1.864
PRIOR_STRENGTH = 3.35e6

def p_fail_lower_bound(a, b, q):
    """Calculates the lower bound of the (1-q) confidence interval for
       a Beta(a, b) distribution."""
    bdist = beta(a, b)
    return bdist.ppf((1.0 - q) / 2.0)

def biod_k(q, p_fail):
    """Calculates the required number of iterations k for BIOD given
       confidence q and failure probability p_fail."""
    return math.ceil(abs(math.log(1.0 - q) / math.log(1.0 - p_fail)))

def biod_run(data, f_transform, f_test, q, p, pass_ratio, timeout_s =
   None, C = DAMPING_CONSTANT, p_s = PRIOR_STRENGTH):
    """
    Runs the BIOD algorithm on the given data.

    Parameters:
        - data: The dataset to run BIOD on.
        - f_transform: Function to transform a set = fn(dataset, seed) -> transformed_dataset
        - f_test: Function to test a set = fn(transformed_dataset) -> bool (pass/fail)
        - q: Desired confidence level (e.g., 0.95 for 95% confidence)
        - p: Pointwise failure probability estimate (should underestimate true failure probability)
        - pass_ratio: Required pass ratio to consider the test successful (e.g., 0.99 for 99% pass rate)
        - timeout: Optional timeout for the test function, in seconds (default: None)
        - C: Damping constant (default: 1.864)
        - p_s: Prior strength (default: 3.35e6)
    """
    start_time = time.monotonic()

    # Calculate damping fraction and adjusted setwise failure probability
    n = len(data)
    damping_fraction = 1.0 / (1.0 + (C * n * p))
    p_prime = p * damping_fraction
    p_fail = 1.0 - (1.0 - p_prime)**n

    # Calculate initial k
    k = biod_k(q, p_fail)
    k_initial = k

    # Initialize priors
    alpha = p_s * p_fail
    beta = p_s * (1.0 - p_fail)

    print(
        f"Starting with initial pass count k = {k_initial}, "
        f"confidence = {q * 100:.2f}%, "
        f"required pass ratio = {pass_ratio * 100:.2f}%"
    )

    iterations = 0
    passes = 0
    unreported_passes = 0
    last_failing_seed = 0
    while iterations < k:
        if iterations - passes > 0 and pass_ratio >= 1.0:
            break

        if timeout_s is not None and (time.monotonic() - start_time) > timeout_s:
```

```python
            print("\nTimeout reached, terminating BIOD run.")
            break

        # Perform transformation and test
        seed = random.getrandbits(64)
        transformed_data = f_transform(data, seed)
        passed = f_test(transformed_data)

        # Record result
        iterations += 1
        if passed:
            passes += 1
            unreported_passes += 1
        elif passes < iterations:
            # Failure detected, update priors and recalculate k
            alpha += 1
            beta += unreported_passes
            unreported_passes = 0

            last_failing_seed = seed
        p_fail = alpha / (alpha + beta)
        k = biod_k(q, p_fail)

    #
    # Final readout
    #
    failures = iterations - passes
    p_fail_bound = p_fail_lower_bound(failures + 1, passes + 1, q)
    print(
        f"{q * 100:.2f}% confident that the true failure rate among datasets is at least {p_fail_bound * 100:.2f}%.\n"
    )

    ratio = (passes / iterations)
    print(f"{failures}/{iterations} tests failed ({ratio * 100.0:.2f}% pass)")

    if iterations < k:
        print(f"BIOD terminated early after {iterations} iterations - not enough evidence to reach required confidence.")
        return False

    if ratio < pass_ratio:
        print(f"BIOD failed to meet required pass ratio of {pass_ratio * 100:.2f}%.")
        return False

    print("BIOD passed successfully.")
    return True
```

Figure and Table Listings