Advanced Processing Methods for Increased Effect Size in Anomalous Cognition, Anomalous Effects and Mind-Matter Interaction

© Scott A. Wilber[1] 2023

Abstract: This paper presents a comprehensive exploration of advanced processing methods, which are crucial for enhancing responsivity in systems designed to utilize mental influence for obtaining non-inferable information and providing hands-free control signals. This exploration encompasses three processing methods – random walk bias amplifier, majority voting, and Bayes' updating. The paper provides detailed descriptions that enable the implementation of a set of sophisticated algorithms, developed to amplify effect size and offer probabilities for accurate assessment of the likelihood of correct results on a per-trial basis.

Keywords: Anomalous Cognition, Anomalous Effects, Mind-Matter Interaction, Responsivity, Effect Size, Signal Processing, Probability Estimation, Majority Voting, Bayes' Updating, Bias Amplification Techniques

Introduction: Random Walk Bias Amplification, Majority Voting and Bayes' Updating

Systems designed to detect or respond to anomalous cognition and anomalous effect, collectively referred to as A.C.E. or mind-matter interaction (*MMI*), process and combine bits from true random number generators. These bits are processed to form trials or blocks of trials that reflect specific desired outcomes. This process uses a number of algorithms which combine a large number of generator bits or trial results into a single output with an increased effect size. Ideally, the output effect size increases by an amount equal to the square root of the number of inputs multiplied by the input bit or trial effect size. However, given the 'quantum-like' nature of A.C.E. and *MMI* (Stapp, 2015), more efficient processing techniques could exist.

This paper examines three algorithms: Random Walk Bias Amplifier (*RWBA*) (Wilber, 2013), Majority Voting (*MV*), and Bayes' Updating (*BU*). Other notable methods like artificial neural networks (*ANN*), rapid machine learning (*RML*) and sequential analysis (Radin, 1990) have been tested, but will not be covered here.

Background

These three methods, *RWBA*, *MV*, and *BU*, yield the same output probability, *Pout*, under certain conditions. This happens when the number of bits or trials used, *N*, is a fixed (odd) number. However, these three methods are fundamentally different in the information they can offer:

- Simple majority voting produces a 1 or a 0 output from a sequence of input bits, but nothing more is known about each sequence or trial result.

[1] President at Core Invention, Inc. swilber@coreinvention.com

- Basic random walk bias amplification also produces a 1 or a 0 output, and in addition the amplification factor ((2 *Pout* – 1)/(2 *p* – 1)) can be calculated and the number of bits needed to reach a bound is measured.
- Bayes' updating produces a 1 or a 0 output, and also provides a probability that the output is the correct state, along with the number of bits or trials needed to reach that probability level.

The efficiency of these algorithms is determined by the extent information in input data is transformed into the final output result. Modeling and simulations of the three methods show majority voting is least efficient and Bayes' updating is essentially on par with random walk bias amplification for high output hit rates.

Here is a description of the three basic algorithms:

1) A basic random walk bias amplifier increases the output probability (*Pout*) as a function of the input probability (*p*) and the number of positions to the bound (*n*):

$$Pout = 1\Big/\left(1 + \left(\frac{1-p}{p}\right)^n\right) \qquad 1.$$

where the two probabilities are the fraction of 1s (or 0s) divided by the total number of bits or trials, assuming *p* > 0.5. The average number of steps to reach a bound (*N*) is:

$$N = n\left(1 - \left(\frac{1-p}{p}\right)^n\right)\Big/\left((2p-1)\left(1 + \left(\frac{1-p}{p}\right)^n\right)\right) \qquad 2.$$

Equations 1 and 2 are derived from analysis of biased bounded random walks (Steele, 2001).

A *RWBA* is a counter that increments one for each 1 input and decrements one for each 0. The counter starts at 0 and terminates when a bound is reached, that is, when the counter reaches a symmetrical positive or negative *n*. If the positive bound is reached, a 1 is output and if the negative bound is reached, a 0 is output. Thereafter the counter is reset to 0 and the process resumes at the next input.

2) Simple Majority Voting counts the number of ones in a sequence of bits or trial results in a series of trials. If the count is greater than one half the number of bits or trials, the result is a 1, otherwise the result is a 0. The probability that the majority vote will provide the correct answer, that is, an absolute majority of correct results, can be calculated using a majority voting equation. The majority voting equation takes as input the user's average per bit or per trial hit rate – previously determined by testing – and the number of bits or trials, *N*, to be combined (majority voted). *N* must be an odd number. The equation calculates the probability the count will be an absolute majority, that is, greater than 50%. The following equation (Scott, 1960) yields the exact probability, *Pout*, of correctly predicting the correct target or outcome given an input sequence of binary guesses (input bits) of length, *N* with probability, *p* (*p* > 0.5):

$$Pout = \sum_{s=a}^{N} p^s (1-p)^{N-s} \binom{N}{s} \qquad 3.$$

where a = Ceiling[(N+1)/2] (rounds the argument to the next higher integer). When N is above a few hundred, this equation becomes unwieldly because of the factorials used in the solution. For large N, *Pout* can be accurately approximated by

$$Pout \cong \text{cdf}(\sqrt{N}(2p-1)) \qquad 4.$$

where cdf is the cumulative distribution function of the normal distribution and $\sqrt{N}(2p-1)$ represents a z-score. A program to calculate **cdf(z)** is in the Appendix.

3) Bayes' Updating calculates the probability the combined results will reveal the correct answer. The input is the user's or users' hit rate(s) and a series of user or users' generator bits or trial results. The probability is updated for each bit or trial result input. The updating is continued until a target probability (a predetermined level of the *posterior* value) is reached. Results from different users with different hit rates can be combined and the resulting *posterior* probability will remain correct as a result of using Bayes' Updating.

Modeling of random walk bias amplification suggests that a sufficiently high generation rate could potentially lead to near 100% hit rates. This projected curve aligns with the actual testing over many orders of magnitude of continually increasing generation rates. However, as hit rates approach 70-80%, even substantial increases in generation rate (up to terabits per second) seem to provide little or no additional increase in effect size (Wilber, 2013).

This divergence might result from a combination of psychological effects and alterations in the entropy source and processing necessary to achieve such high generation rates. Users often struggle with the concept of achieving very high hit rates through mental influence. This is illustrated by the user who starts a test series and scores 10 consecutive hits, with a common response being to stop and say, "the tester is broken." Helmut Schmidt (1982), in one of the "observation causes collapse" theories, suggests there is a quantum mechanical limit of 75% hit rate for single trials.

The method of Bayes' updating presents a viable solution to overcome the limitations observed in 'brute force' bias amplification approaches. *BU* allows the integration of individual trials, each with a relatively modest hit rate. Bayes' updating can be a stand-alone processing method or use results from other processing methods to attain very high target probabilities. William Jefferys (1990) discusses the strengths of Bayesian Analysis versus frequentist statistical tests when applied to *MMI* results.

Advanced Processing

Advanced processing methods provide two important advantages:

- Increased responsivity reflected in higher hit rate or effect size. This allows useful results to be obtained much more quickly and with less user effort.
- Figure of merit to assess accuracy, usually in the form of a probability or surprisal value. This provides a way to give user feedback on a per-trial basis and determine an objective confidence level for results.

Advanced random walk bias amplifier (RWBA) algorithm

The basic random walk bias amplifier is an algorithm for increasing bias in an output bitstream while reducing the number of bits. The information contained in the input in the form of bias is maintained with almost 100% efficiency in the output bias. It's a way to use high-speed random generators without having to transfer and process up to billions of bits per second.

Advanced processing uses properties of the *RWBA* to increase responsivity, which is defined as increased hit rate (hits/trials) or effect size (2 x hit rate –1), and provide a measurement of accuracy through the null hypothesis probability (*p*-value). In this context, a lower *p*-value is associated with higher accuracy on a trial-by-trial basis.

The following steps are based on a 100kbps generator rate and a nominal trial duration of 201ms:

1) A basic RWBA is used to produce 21 subtrials using an average of 961 input bits each, with bounds at $n = \pm 31$. The basic algorithm is a counter starting at 0. The input bits are converted to ± 1 by multiplying by 2 and subtracting 1. These bits are added to the total in the counter until the count reaches one of the bounds, either +31 or –31. Then an output result is produced: +1 if the positive bound is reached and –1 if the negative bound is reached. At the same time a count is kept of the number of input bits (N) used to reach either bound. Note, N is always the same parity as n (both odd in this case).
2) In addition to the subtrial output results, a weighting factor is calculated for each subtrial. The idea underlying these weights is that the drift rate (the rate at which the walker moves toward a bound) is equal to the effect size of mental influence.[2] The higher the effect size, the fewer the number of steps needed to reach a bound. The probability (p) of each subtrial occurring by chance is calculated from the cumulative distribution function (CDF) as a function of N.
   The subtrial CDF is found from the empirical probability mass function (PMF) of the results of 4 million simulated RWBA outputs with $n = 31$. The CDF is the integral of the PMF. Finally, a polynomial curve fit is found for the CDF using Ln(N) as the argument. The curve fit is valid for $107 \leq N \leq 4391$ ($0.0044 \leq p \leq 0.9956$). To avoid extreme outliers, the range of p values is limited to $0.0625 \leq p \leq 0.840896$ by limiting the range of N. This limits the range of subtrial weights is 16 to 1, so a single outlier cannot easily overwhelm the measurement.

[2] In a biased random walk, the drift rate is $p - q$, where p and q are the probability of moving toward and away from the bound respectively. Effect size is the hit rate – the miss rate. These two definitions are equivalent when the bias is caused by mental influence.

The curve fit of the CDF is:
If $N < 209$, return $p = 0.0625$, else if $N > 1613$, return $p = 0.840896$, else
x = Ln(N)
p = - 14753.24169815 + 27287.435642795x - 22758.051790793x^2 + 11296.73926749x^3 -3708.2656031604x^4 + 845.26673369466x^5 - 136.52944311971x^6 + 15.628082615014x^7 -1.2424970617642x^8 + 0.065349290419121x^9 - 0.0020465302532178x^10 + 0.000028912696259115x^11
Test results: cdf(403) = 0.245676, cdf(971) = 0.635516

3) Each subtrial weight is determined by the surprisal value (*SV*), which is calculated from its specific *p*-value, after any defaults have been applied. We use the surprisal value because it provides a more uniform range of values for weights.
   The surprisal value is given by:
   $SV = \text{Log}_2(1/p) = \text{Ln}(1/p)/\text{Ln}(2)$ 5.
4) Calculate 21 weighted subtrial values by multiplying each subtrial output (+1 or –1) by its surprisal value.
5) Calculate a normalized weighted trial value (nwtv). This is done by adding together the weighted values of the 21 subtrials and dividing the sum by the total of the 21 weights (*SV*s). In equation form: nwtv = (Sum of the 21 weighted subtrials) / (Sum of the 21 weights). The resulting nwtvs are floating-point numbers between –1.0 and +1.0. A positive nwtv indicates a bias toward more 1s, whereas a negative nwtv indicates a bias towards more 0s.
6) The next step is to calculate the *p* value, which will depend on the type of test being conducted: one-tailed or two-tailed. This distinction hinges on whether a target direction or ‘aim’ is specified.

   - A one-tailed test is conducted when an aim is set to produce either more 1s or more 0s, providing a specified direction.
   - A two-tailed test is used when no specific direction is indicated.

   Typically, user-initiated trials are one-tailed, and continuous trials are two-tailed for control applications. However, even continuous trials can be one-tailed for testing purposes.

   For a one-tailed test:

   - If the aim is to produce more 0s, the trial *p* value is derived from the cdf(nwtv) function.
   - If the aim is to produce more 1s, the trial *p* value is calculated as 1 – cdf(nwtv).

   For a two-tailed test (when no specific aim is set):

   - If the nwtv is negative, the trial *p* value is calculated as 2 x cdf(nwtv).
   - If the nwtv is positive, the trial *p* value is 2 x (1 – cdf(nwtv)).

When calculating *p* from the nwtv CDF (cdf(nwtv)), a limit is set. If nwtv < –0.95, set nwtv = –0.95. If nwtv > 0.95, set nwtv = 0.95. If nwtv is within these bounds, it remains unchanged. The equation to calculate **cdf(nwtv)** is in the Appendix.

The statistical significance of the trial depends on the *p* value of the null hypothesis, which represents the probability that the observed result could have happened by chance, absent any mental influence. If $p < 0.05$, the result is deemed statistically significant, rejecting the null hypothesis at the observed probability level.

The following steps are used to produce blocks of consecutive trials or data 'windows' when results from longer duration tests are desired.

1) For a trial duration longer than 201ms, such as one second, the results of 5 consecutive 201ms trials are combined to form what is referred to as a window. This process involves the following steps:

   - Take the *p* values for each of the 5 trials. Remember, these *p* values should be calculated without incorporating any adjustments for one- or two-tailed results. This prevents bias when combining results from multiple trials.
   - Convert each of these *p* values into a z-score using the provided conversion function.
   - Sum up the converted z-scores.
   - Divide this sum by the square root of 5, which is the number of trials being combined.
   - Finally, convert this resulting average z-score back into a probability using the CDF of the standard normal distribution.

   Mathematica programs to convert probabilities (*p*) to z-scores, **invnorm(p)**, and to convert z-scores to probabilities (*p*), **cdf(z)**, are found in the Appendix.

   For windows longer than 1 second, increase the number of 201ms trials used. However, be aware that user feedback, if provided, will not be real-time for these extended duration windows. The trial *p* values are always converted to z-scores for combination into a window z-score, and then into a window *p* value. Trial *p* values are not adjusted as one- or two-tailed unless the trial is the final output. This means that one- or two-tailed adjustments are only applicable for the final result, not for intermediary computations during the combination of trials.
2) The calculation of the *p* value for window data is outlined in step 1. However, because the window data can usually result in either a +1 or –1 outcome, the result is treated as two-tailed:

   - If the *p* value is less than 0.5, double it. This becomes the window *p* value and the window result is interpreted as –1.
   - If the *p* value is equal to or greater than 0.5, calculate $2(1 - p)$. This becomes the window *p* value and the window result is interpreted as +1.

Even though a one-tailed test is theoretically feasible for window results under specific testing scenarios, the two-tailed test is generally applicable in continuous and hands-free control applications. In that context, results of +1 or –1 are equally likely. For instance, you may wish to either turn a light on (+1) or off (–1) in such an application.

3) To calculate the *p* value for a series of 201ms trials, or for a series of windows, use the Binomial distribution. This probability distribution provides the probability of getting X or more hits (or "successes") in a given number of trials, with the probability of success in each trial being 0.5.

   A Mathematica program to calculate the Binomial probability, *p* (X ≥ number of hits), **binomialp**, is found in the Appendix.

Summary of Steps for Generating Trials:

1) Use the basic random walk bias amplifier (RWBA) method to produce 21 subtrials with a bound of *n* = 31using an average of *N* = 961 input bits each.
2) Calculate a weight for each subtrial using the cumulative distribution function (CDF) of the number of bits to reach a bound.
3) Calculate the Surprisal Value (*SV*) for each subtrial from its specific *p*-value. This is used to create a weighted output for each subtrial.
4) Calculate a normalized weighted trial value (nwtv) by summing all the weighted subtrial values and dividing by the sum of the weights. The sign of the nwtv indicates the bias direction.
5) Calculate the *p*-value for the trial. If there is a specified direction or 'aim', use a one-tailed test. If no aim is specified, use a two-tailed test.
   - For one-tailed tests, calculate the *p*-value using the cdf(nwtv) function, which varies depending on the aim.
   - For two-tailed tests, calculate the *p*-value based on whether the nwtv is negative or positive.
6) The statistical significance of the trial based on the *p*-value of the null hypothesis.
7) Use the Binomial distribution to calculate the *p*-value for a series of trials. This provides the probability of achieving more than or equal the number of hits in the number of trials with a success probability of 0.5.

Summary of Steps for Generating a Window from multiple trials:

1) For trials longer than 201ms, combine the results of multiple consecutive trials to form a 'window'.
2) Take the *p*-values for each of the trials (without any one- or two-tailed adjustments) and convert them into z-scores.
3) Divide the sum the z-scores by the square root of the number of trials being combined.
4) Convert the resulting average z-score back into a probability using the CDF of the standard normal distribution.
5) Convert the window *p*-value to a two-tailed probability. If the *p*-value is less than 0.5, double it; if it's equal to or greater than 0.5, calculate 2(1 – *p*).

6) Use the Binomial distribution to calculate the *p*-value for a series of windows. This provides the probability of achieving more than or equal the number of hits in the number of trials with a success probability of 0.5.

Advanced majority vote (*MV*) processing algorithm

The basic majority vote method is an algorithm for increasing bias in an output bitstream while reducing the number of bits. Its overall efficiency is not as high as a random walk bias amplifier, but it has an advantage in that an output is produced with a fixed number of input bits.

Advanced processing uses statistical properties of the *MV* to increase responsivity, which is defined as an increased hit rate (hits/trials) or increased effect size (2 x hit rate –1), and provide a measurement of accuracy through the null hypothesis probability (*p*-value). In this context, a lower *p*-value is associated with higher accuracy on a trial-by-trial basis.

The following steps are based on a 100kbps generator rate and a nominal trial duration of 201ms:

1) A basic majority vote is used to produce 21 subtrials using 961 input bits (*N*) each. The basic algorithm counts the number of 1s in a sequence. An odd number of input bits is used to prevent ties in the counts, which would be an indeterminate result. If the count of 1s is > *N*/2, then the output result is a +1, or else the output result is a –1.
2) In addition to the subtrial output results, a weighting factor is also calculated for each subtrial. The weights are based on an unconventional use of the *p* value, where a lower *p* value of the null hypothesis suggests an increased likelihood of the result being due to mental influence.
   The probability (*p*) of each subtrial occurring by chance is calculated from the cumulative distribution function (CDF) of the counts of 1s. The counts for each subtrial follow a normal distribution with a mean of 480.5 ($\mu = N/2$) and a standard deviation of 15.5 ($\sigma = \text{sqrt}(N/4)$), where $N = 961$ is the total number of input bits for the subtrial. First, the counts of 1s are standardized to create a z-score: z = (count of 1s – μ)/ σ. Then, the cdf[z_] function, as previously defined, is used to calculate the *p* value for each subtrial. The calculated *p* values are then adjusted based on the output result of each subtrial: if the output result is –1, the *p* value is adjusted to 2 x cdf(z); if the output result is +1, the *p* value is adjusted to 2 x (1 – cdf(z)). To avoid extreme outliers, the range of *p* values is limited: if $p < 0.0625$, then *p* is set to 0.0625; if $p > 0.840896$, then *p* is set to 0.840896.
3) Each subtrial weight is determined by the surprisal value (*SV*), which is calculated from its specific *p*-value, after any defaults have been applied. We use the surprisal value because it provides a more uniform range of values for weights.
   The surprisal value is given by:
   $SV = \text{Log}_2(1/p) = \text{Ln}(1/p)/\text{Ln}(2)$.
4) Calculate 21 weighted subtrial values by multiplying each subtrial output result (+1 or –1) by its surprisal value.

5) Calculate a normalized weighted trial value (nwtv). This is done by adding together the weighted values of the 21 subtrials and dividing the sum by the total of the 21 weights (*SV*s). In equation form: nwtv = (Sum of the 21 weighted subtrials) / (Sum of the 21 weights). The resulting nwtvs are floating-point numbers between –1.0 and +1.0. A positive nwtv indicates a bias toward more 1s, whereas a negative nwtv indicates a bias towards more 0s.
6) The next step is to calculate the *p* value for the trial, which will depend on the type of test being conducted: one-tailed or two-tailed. This distinction hinges on whether a target direction or 'aim' is specified.

   - A one-tailed test is conducted when an aim is set to produce either more 1s or more 0s, providing a specified direction.
   - A two-tailed test is used when no specific direction is indicated.

   Typically, user-initiated trials are one-tailed, and continuous trials are two-tailed for control applications. However, even continuous trials can be one-tailed for testing purposes.

   For a one-tailed test:

   - If the aim is to produce more 0s, the trial *p* value is derived from the cdf(nwtv) function.
   - If the aim is to produce more 1s, the trial *p* value is calculated as 1 – cdf(nwtv).

   For a two-tailed test (when no specific aim is set):

   - If the nwtv is negative, the trial *p* value is calculated as 2 x cdf(nwtv).
   - If the nwtv is positive, the trial *p* value is 2 x (1 – cdf(nwtv)).

   When calculating *p* from the nwtv CDF (cdf(nwtv)), a limit is set. If nwtv < –0.95, set nwtv = –0.95. If nwtv > 0.95, set nwtv = 0.95. If nwtv is within these bounds, it remains unchanged. The equation to calculate **cdf(nwtv)** is in the Appendix.

   The statistical significance of the trial depends on the *p* value of the null hypothesis, which represents the probability that the observed result could have happened by chance, absent any mental influence. If $p < 0.05$, the result is deemed statistically significant, rejecting the null hypothesis at the observed probability level.

The following steps are used to produce blocks of consecutive trials or data 'windows' when results from longer duration tests are desired.

1) For a trial duration longer than 201ms, such as one second, the results of 5 consecutive 201ms trials are combined to form what is referred to as a window. This process involves the following steps:

- Take the *p* values for each of the 5 trials. Remember, these *p* values should be calculated without incorporating any adjustments for one- or two-tailed results. This prevents bias when combining results from multiple trials.
- Convert each of these *p* values into a z-score using the provided conversion function.
- Sum up the converted z-scores.
- Divide this sum by the square root of 5, which is the number of trials being combined.
- Finally, convert this resulting average z-score back into a probability using the CDF of the standard normal distribution.

Mathematica programs to convert probabilities (*p*) to z-scores, **invnorm(p)**, and to convert z-scores to probabilities (*p*), **cdf(z)**, are found in the Appendix.

For windows longer than 1 second, increase the number of 201ms trials used. However, be aware that user feedback, if provided, will not be real-time for these extended duration windows. The *p*-values of the trials are always converted to z-scores before being combined into a window z-score, which is then converted into a window *p*-value. Trial *p* values are not adjusted as one- or two-tailed unless the trial is the final output. This means that one- or two-tailed adjustments are only applicable for the final result, not for intermediary computations during the combination of trials.

2) The calculation of the *p* value for window data is outlined in step 1. However, because the window data can usually result in either a +1 or –1 outcome, the result is treated as two-tailed:

   - If the *p* value is less than 0.5, double it. This becomes the window *p* value and the window result is interpreted as –1.
   - If the *p* value is equal to or greater than 0.5, calculate 2(1 – *p*). This becomes the window *p* value and the window result is interpreted as +1.

   Even though a one-tailed test is theoretically feasible for window results under specific testing scenarios, the two-tailed test is generally applicable in continuous and hands-free control applications. In that context, results of +1 or –1 are equally likely. For instance, you may wish to either turn a light on (+1) or off (–1) in such an application.

3) To calculate the *p* value for a series of 201ms trials, or for a series of windows, use the Binomial distribution. This probability distribution provides the probability of getting X or more hits (or “successes”) in a given number of trials, with the probability of success in each trial being 0.5.

   A Mathematica program to calculate the Binomial probability, *p* (X ≥ number of hits), **binomialp**, is found in the Appendix.

Summary of Steps for Generating Trials:

1) Use the basic Majority Vote (MV) method to produce 21 subtrials using 961 input bits each.
2) Calculate a weight for each subtrial using the cumulative distribution function (CDF) of the counts of 1s. The counts should follow a normal distribution with a mean of 480.5 and a standard deviation of 15.5.
3) Calculate the Surprisal Value (*SV*) for each subtrial from its specific *p*-value. This is used to create a weighted output for each subtrial.
4) Calculate a normalized weighted trial value (nwtv) by summing all the weighted subtrial values and dividing by the sum of the weights. The sign of the nwtv indicates the bias direction.
5) Calculate the *p*-value for the trial. If there is a specified direction or 'aim', use a one-tailed test. If no aim is specified, use a two-tailed test.
   - For one-tailed tests, calculate the *p*-value using the cdf(nwtv) function, which varies depending on the aim.
   - For two-tailed tests, calculate the *p*-value based on whether the nwtv is negative or positive.
6) The statistical significance of the trial based on the *p*-value of the null hypothesis.
7) Use the Binomial distribution to calculate the *p*-value for a series of trials. This provides the probability of achieving more than or equal the number of hits in the number of trials with a success probability of 0.5.

Summary of Steps for Generating a Window from multiple trials:

1) For trials longer than 201ms, combine the results of multiple consecutive trials to form a 'window'.
2) Take the *p*-values for each of the trials (without any one- or two-tailed adjustments) and convert them into z-scores.
3) Divide the sum the z-scores by the square root of the number of trials being combined.
4) Convert the resulting average z-score back into a probability using the CDF of the standard normal distribution.
5) Convert the window *p*-value to a two-tailed probability. If the *p*-value is less than 0.5, double it; if it's equal to or greater than 0.5, calculate $2(1 - p)$.
6) Use the Binomial distribution to calculate the *p*-value for a series of windows. This provides the probability of achieving more than or equal the number of hits in the number of trials with a success probability of 0.5.

Bayes' updating (*BU*) algorithm

Bayesian analysis is a robust mathematical tool that utilizes probability theory to combine the outcomes of multiple mental efforts in Anomalous Cognition and Effect (A.C.E.) and Mind-Matter Interaction (*MMI*) systems. This sophisticated approach is particularly useful when addressing challenging tasks, like predicting future events, which cannot be deduced from existing information. This section outlines a processing method that utilizes Bayes' updating to combine multiple bits or trials, yielding a single bit of information with an associated probability of correctness.

Bayesian analysis primarily employs three types of probabilities:

1) Joint probability refers to the probability of two events, *A* and *B*, occurring simultaneously. A number of notations, which are all equivalent, represent this probability:

$P(A \,\&\, B)$, $P(A \cap B)$, $P(A \text{ and } B)$, $P(A, B)$ or $P(A \wedge B)$

2) Conditional probability refers to the probability of an event *A* occurring, given that another event *B* has already occurred, represented as

$P(A \mid B)$ or $P_B(A)$

3) Marginal probability is the standalone probability of a single event, *B*, occurring irrespective of the occurrence or non-occurrence of any other events. The sum of all marginal probabilities must equal 1.0, depicted as

$P(B)$

A number of commonly used variable names are used in the presented solution.

1) *Likelihood*, in this context, is the average probability of achieving the correct result from a measurement. Likelihood is equivalent to the user's hit rate (*HR*) based on bits, trials or windows. The hit rate will vary depending on the user and the nature of the task, and it is assessed and updated based on empirical results. HR is expected to be ≥ 0.5: results opposite to the intended direction or $HR < 0.5$ are deemed neither valid or useful.
2) *Prior* denotes the current belief or probability that a particular state or answer is correct. In the absence of other information, the initial prior is typically set at 0.5 (uninformative prior).
3) *Posterior* denotes the believed probability that a certain state or answer is correct after updating it with an observation – either a bit, a trial, or a window of trials. Once the posterior is computed, its value is used as the prior for the next update.
4) An observation (referred to as *obs*) is the outcome of a mental effort to produce a result. An *obs* can be either a 1 or a 0, but these symbols generally signify an event or condition. The symbol '1' could represent up, higher, larger, stronger, faster, more, etc., while '0' would represent the opposite; down, lower, smaller, weaker, slower, less, etc. For optimal results, these symbols should correlate closely with the concept they represent. In situations where there is no clear or obvious association, the best logical guess is applied. For instance, '1' could be associated with outer, even, white, and hot, whereas '0' could

represent inner, odd, black, and cold. With practice, users can learn to reliably associate specific symbols with particular variables or outcomes.

5) *T* and *F* are abbreviations for True and False. They are used in various equations to indicate a variable or state is either true or false.

Simple Bayes' Updating in A.C.E. and MMI Applications

In its simplest form, an application based on measurements of mental influence to obtain information or provide control signals deals with binary or two-state systems, i.e., 1 or 0. The key question is determining which state is the most likely to be true and how likely it is.

Derivation:

Initial State
Initially, we don't have definitive information to favor one state over the other. Hence, we assign an equal or "uninformative" *prior* probability of 0.5 to both states. This implies a 50/50 chance that either '1' or '0' is the correct answer.

Hit Rate
A key parameter in this context is the Hit Rate (*HR*), which represents the *likelihood* of correctly predicting a given state, based on previous experience or default estimates. Hit rate will vary for different users and tasks. It's possible for a user's *HR* to be different for states 1 and 0 (denoted as HR_1 and HR_0). If known, these different *HR*s can be incorporated into the Bayesian updating model. The derivation of Bayes' updating equations using different hit rates for 1s and 0s, **When Hit Rates for 1s and 0s differ**, is found in the Appendix.

Bayesian Updating and Bayes' Rule
Bayesian updating is a process used in Bayesian inference to refine a probability estimate for a hypothesis as more evidence or data becomes available. This process is based on Bayes' Rule.

The mathematical form of Bayes' rule is:

$$P(A \mid B) = \frac{P(B|A) * P(A)}{P(B)}$$

Where:

- $P(A \mid B)$ is the posterior probability of hypothesis *A* given the evidence *B*.
- $P(B \mid A)$ is the likelihood of observing the evidence *B* given that hypothesis *A* is true.
- $P(A)$ is the prior probability of hypothesis *A* being true before observing the evidence.
- $P(B)$ is the total probability of the evidence (observed across all different hypotheses).

For our applications *A* refers to a particular state being true, and *B* refers to the observation or result of a bit, a trial or a window of trials.

Updating Process

After each trial or observation, we apply Bayes' Rule to update our probabilities, turning the *prior* probability into a *posterior* probability. Different equations are used to update the probability of 1 = *T* based on whether the observation is a 1 or a 0. The *likelihood* also depends on the observation: if *obs* = 1, the likelihood is *HR*, while if *obs* = 0, the likelihood is (1 – *HR*). Here are the equations and a brief derivation:

When the observation or *obs* is 1, calculate the total probability of the observation being a 1, *P*(*obs* = 1) and update the posterior probability:

1) Calculate the joint probability of the state being 1 and the observation being 1. This represents the probability of correctly predicting a 1:
   $P(1 = T \,\&\, obs = 1) = P(1 = T) * P(obs = 1|1 = T) = prior * \ HR$
2) Calculate the joint probability of the state being 0 and the observation being 1, which represents the probability of incorrectly predicting a 1:
   $P(1 = F \,\&\, obs = 1) = P(1 = F) * P(obs = 1|1 = F) = (1 - prior) * \ (1 - HR)$
3) Sum these two results to get the total probability of observing a 1:
   $P(obs = 1) = prior * \ HR + (1 - prior) * \ (1 - HR)$
4) Use Bayes' Rule to update the posterior probability for $P(1 = T \mid obs = 1)$, which is the probability that '1' is the correct state or answer, given that a 1 was observed:
   $$P(1 = T \mid obs = 1) = \frac{HR * \ prior}{HR * \ prior + \ (1 - HR) * (1 - prior)}$$

When *obs* is 0, calculate the total probability of the observation being a 0, *P*(*obs* = 0) and update the posterior probability:

1) Calculate the joint probability of the state being 1 and the observation being 0, which represents the probability of incorrectly predicting a 0:
   $P(1 = T \,\&\, obs = 0) = P(1 = T) * P(obs = 0|1 = T) = prior * \ (1 - HR)$
2) Calculate the joint probability of the state being 0 and the observation being 0, which represents the probability of correctly predicting a 0:
   $P(1 = F \,\&\, obs = 0) = P(1 = F) * P(obs = 0|1 = F) = (1 - prior) * \ HR$
3) Sum these two results to get the total probability of observing a 0:
   $P(obs = 0) = prior * \ (1 - HR) + \ (1 - prior) * HR$
4) Use Bayes' Rule to update the posterior probability for $P(1 = T \mid obs = 0)$, which is the probability that '1' is the correct state or answer, given that a 0 was observed:
   $$P(1 = T \mid obs = 0) = \frac{(1 - HR) * prior}{(1 - HR) * prior + \ HR * (1 - prior)}$$

Once a new *posterior* is calculated, it becomes the *prior* for the next trial, reflecting our updated belief about the state of the system.

These equations are used when updating the hypothesis that a ‘1’ is the true state. A posterior less than 0.5 indicates a ‘0’ is the true state and its probability of being correct is $P(0 = T) = 1 -$ *posterior*.

The following Mathematica program implements the solution found above and calculates the posterior taking as inputs, *likelihood* (the user’s *HR*), *prior* and *obs* (1 or 0):

```
(* Function to calculate the posterior probability *)
posterior[likelihood_, prior_, obs_] :=
   If[obs == 1,
     (* Case when obs=1 *)
     (likelihood * prior) / (prior * likelihood + (1 - prior) * (1 - likelihood)),
     (* Case when obs=0 *)
     ((1 - likelihood) * prior) / (prior * (1 - likelihood) + (1 - prior) * likelihood)
   ]
```

The likelihood, synonymous with the user’s hit rate (*HR*), generally stays constant throughout a series of updates, unless multiple users with differing hit rates contribute to the final result. After each iteration, the calculated posterior probability will become the prior probability for the next update. Following each update, we identify the state (either ‘1’ or ‘0’) that has the highest probability of being correct based on the current posterior. Specifically, if the posterior probability is greater than 0.5, then ‘1’ is deemed the most probable state; if it’s less than 0.5, then ‘0’ is considered the most probable, and its probability is 1 – *posterior*.

Analysis of the effect of using inaccurate Hit Rates in Bayes’ Updating.

Accurately measuring a user’s hit rate may not always be possible, or the hit rate may vary depending on the task at hand. Thus, simulation can be used to estimate the consequences of over- or under-estimating the actual hit rate (*likelihood*). An examination of three parameter sets provides insights:

1) With a nominal *HR* = 0.52 and a target probability of 60%, a ±10% variation in the actual effect size (*HR*s = 0.522 and 0.518) did not notably impact the number of trials or the actual probability reached.
2) For a nominal *HR* = 0.54 and a target probability of 80%, actual effect sizes of 0.12 and 0.0533 (*HR*s = 0.56 and 0.5267) resulted in 60 and 76 trials respectively to reach the target probability. The actual probabilities reached were approximately 89.7% and 72.2%.
3) With the same nominal *HR* as in (2) but a target probability of 95%, the numbers of trials to reach the target probability were 155 and 273 respectively. The actual probabilities reached were around 99% and 88.3%.

When the estimated hit rate approximates the actual hit rate, the outcomes are relatively unchanged. Underestimating the hit rate results in reaching the target with fewer trials than expected and exceeding the target probability, both non-detrimental outcomes. Overestimation, however, results in a higher number of trials than expected and a less reliable outcome.

These findings suggest that a conservative estimate of the hit rate is preferable to an overestimation. For critical applications, one strategy to mitigate overestimation is to set a higher-than-necessary target probability.

While the models and equations provided here have been confirmed by extensive simulations, it's important to note that the actual effects of mental influence may deviate from mathematical models. Nevertheless, modeling and simulation provide a valuable starting point for real-world testing.

In this example of Bayes' updating, the aim is to determine the correct result out of two possible states, represented by '1' or '0'. After each update, a threshold is calculated as the maximum of the posterior and 1-posterior. The series stops when the threshold exceeds or equals the target probability. The resulting answer corresponds to '1' or '0', depending on whether the posterior or 1 – posterior was the larger number, representing the correct event or condition.

How many trials are needed to reach a target probability of being correct?

The posterior value is the probability that the state inferred through the Bayes' Updating is correct, given the *likelihood* or hit rate provided. Figure 1 shows the average number of trials required to reach the target *posterior* value. Comparable results using basic random walk bias amplification and majority voting are shown for comparison. For this plot, the hit rate is 0.52 (*ES* = 0.04). Depending on the task and the user, the actual hit rate may be higher or lower.

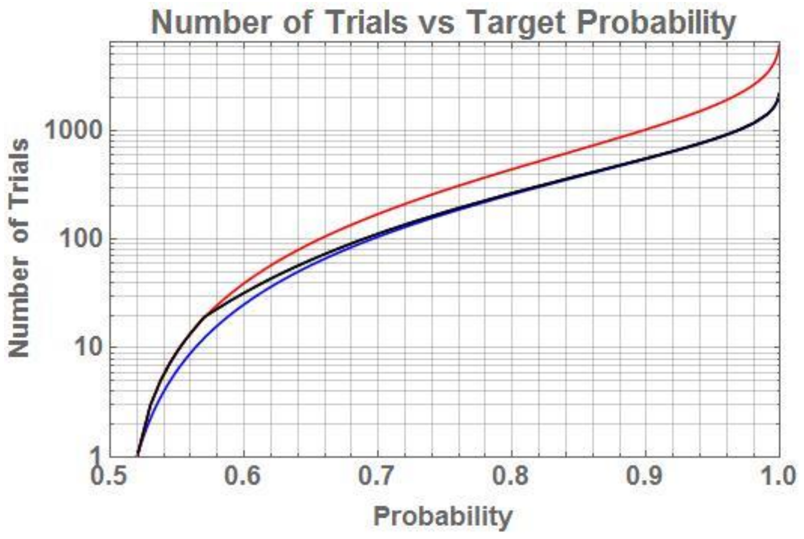


Figure 1

The black line is the number of trials using Bayes' Updating (*BU*), the blue line (bottom) is the number indicated from random walk bias amplification (*RWBA*) and the red line (top) is calculated using the majority voting (*MV*) probability equation. The number of trials for *BU* are bounded above by the *MV* and below by *RWBA*: $RWBA \leq BU \leq MV$.

The black line in Figure 1 shows that Bayes' Updating takes a number of *extra* trials – about 7 – to start converging toward the *RWBA* curve because the initial *prior* or starting point is assumed to be 0.5. These trials are needed to move the *posterior* toward the true underlying value. If the initial prior is set to the hit rate, this discrepancy disappears, but in real applications there is no reason to assume the result will be a 1 or a 0.

An important confirmation from these models and simulations is that the fraction of thousands of series of trials that reach the expected outcome is equal to the target probability. That is, if the target probability is 0.95, 95% of series of trials provide the correct result, and the false negative rate is, 1 – positive rate = 0.05. The odds are 19:1 of reaching a correct result. If the target probability is set as low as 0.667, the odds of getting a correct result are only 2:1 – potentially inadequate for many practical applications. It seems likely a target probability of at least 0.80, giving 4:1 odds of "winning," is a reasonable starting point.

It's interesting to observe from the models that the number of trials needed to reach a high level of probability or expected accuracy does not increase rapidly. For $p = 0.999999$, $N = 4322$, only twice as many trials as needed for $p = 0.999$. To put this in perspective, consider a scenario with a hundred skilled users contributing about 43 trials in only a few minutes, which could provide one bit of information with virtual certainty.

The model also confirms that the number of trials required to reach a particular target probability is very sensitive to the user's skill level, specifically the hit rate. The number of trials is inversely proportional to the square of the effect size. That means if it takes 267 trials to reach a target probability of 80% at a hit rate of 0.52 (effect size = 0.04), it will take 4166 trials, about 16 times as many, to reach the 80% target if the hit rate is 0.505 (effect size = 0.01). On the other hand, if the hit rate is 0.54, only about 72 trials will be needed. This relationship makes it clear why using the most responsive hardware and processing algorithms is so critical.

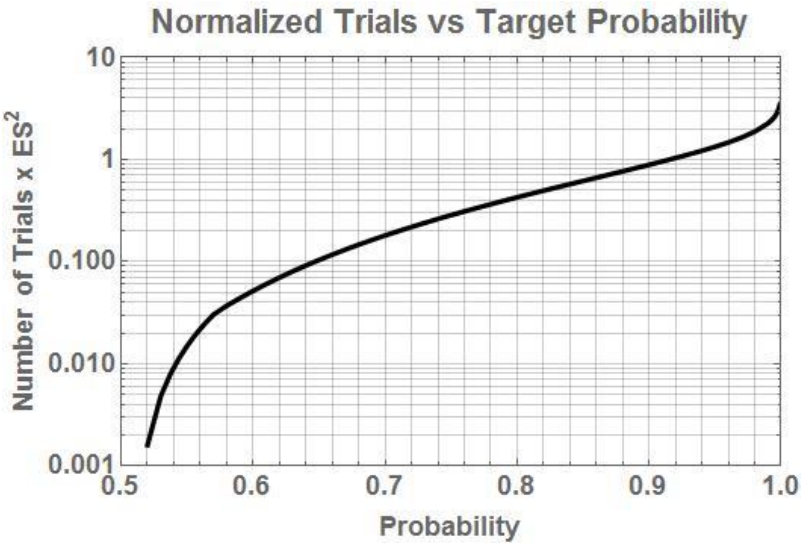


Figure 2

The number of trials (N) can be normalized by multiplying N by ES squared, $Ntrial = N \times ES^2$. Figure 2 shows the normalized number of trials ($Ntrial$) versus target probability. Achievable levels of Hit Rate for single trials span the range from 0.505 to 0.58 (ES = 0.01 to 0.16). The average number of trials, N, is recovered by dividing by effect size squared, $N = Ntrial/ES^2$.

Conclusion

This paper presents comprehensive descriptions of methods developed to enhance effect size and responsivity in systems that harness Anomalous Cognition, Anomalous Effects (A.C.E.) or Mind-Matter Interaction (MMI). The proposed advanced algorithms are designed to amplify effect size, while simultaneously providing probabilities that allow evaluation of results on a trial-by-trial basis.

These state-of-the-art algorithms serve as essential tools for researchers and designers committed to optimizing the performance of mind-enabled systems. Future research should continue to explore these and other innovative methods to enhance results and validate the techniques elucidated in this paper.

While this paper constitutes another significant stride in our ongoing research, there is still much to learn and understand. We hope that this work will act as a catalyst for continued exploration, ultimately leading to expanded utility and effectiveness in real-world applications.

References

Jefferys, W. H. (1990). Bayesian Analysis of Random Event Generator Data. *Journal of Scientific Exploration*, 4(2), 153–169. http://quasar.as.utexas.edu/papers/reg.pdf

Radin, D. I. (1990-91). Enhancing Effects in Psi Experiments with Sequential Analysis: A Replication and Extension. *European Journal of Parapsychology*, 8, 98–111. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=090C663E8629733CE0F3ABDA8B3DC0E1?doi=10.1.1.138.1436&rep=rep1&type=pdf

Schmidt, H. (1982). Collapse of the state vector and psychokinetic effect. *Foundations of Physics*, 12, 565–581. https://ia600307.us.archive.org/20/items/NotesonSpiritualismandPsychicalResearch/SchmidtcollapseoftheStatevector.pdf

Scott, C. (1960). An Appendix to the Repeated Guessing Technique. *International Journal of Parapsychology*, 2(3), 37–45.

Stapp, H. P. (2015). A quantum-mechanical theory of the mind-brain connection. In *Beyond Physicalism*. E.F. Kelly, et al. (Eds.), (pp. 157–193). Lanham: Rowman and Littlefield.

Steele, J. M. (2001). Random Walks and First Step Analysis. In: *Stochastic calculus and financial applications*. I. Karatzas & M. Yor (Eds.), (pp. 1–9). Springer-Verlag New York, NY.

Wilber, S. A. (2013). Advances in Mind-Matter Interaction Technology: Is 100 Percent Effect Size Possible? https://coreinvention.com/wp-content/uploads/2022/08/Advances-in-Mind-Matter-Interaction.pdf

Appendix

cdf(nwtv)

```
cdf[nwtv_]:= (x = -Abs[nwtv];
xx = 1.3671649364575 x + 0.043433149991109 x^2 - 2.16454907883120 x^3 -
1.16398609859974 x^4 - 15.9478516592348 x^5 - 86.4404062808434 x^6 - 201.9161410163
x^7 - 265.2908149166 x^8 - 205.91445301453 x^9 - 88.495808283824 x^10 -
16.271768076703 x^11;
If[nwtv < 0, p = .5 + xx, p = .5 - xx]) (*The function is symmetrical to prevent bias.*)
```

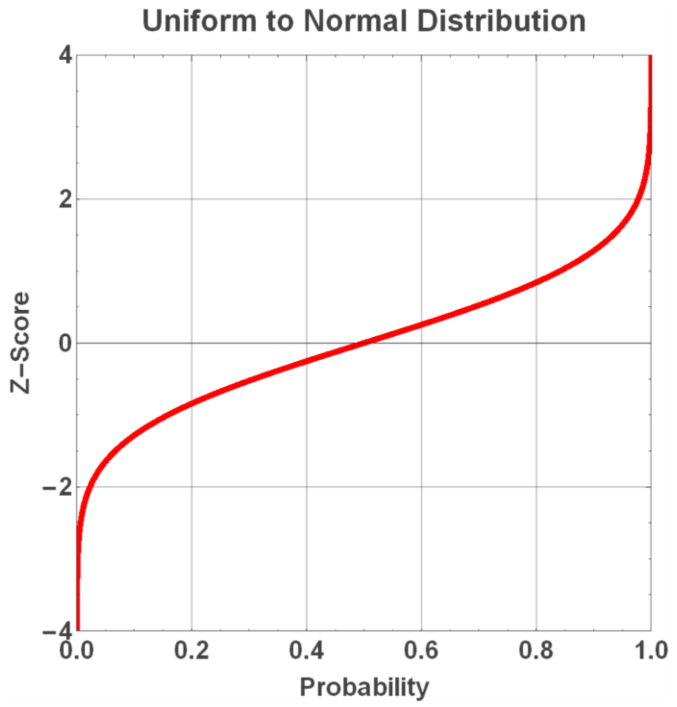
Test values are: input –0.244, *p* = 0.19949898165; input 0.700, *p* = 0.99551358328

invnorm(*p*)

The following Mathematica equation converts probabilities (*p*) to z-scores:

```
invnorm[p_] := (p0 = -.322232431088; p1 = -1.0; p2 = -.342242088547; p3 = -.0204231210245;
p4 = -.453642210148 10^-4; q0 = .099348462606; q1 = .588581570495; q2 = .531103462366;
q3 = .10353775285; q4 = .38560700634 10^-2; (*program constants*)
If[p < .5, pp = p, pp = 1. - p];
y = Sqrt[Log[1/(pp^2)]];
xp = y + ((((y p4 + p3) y + p2) y + p1) y + p0)/((((y q4 + q3) y + q2) y + q1) y + q0);
If[p < .5, xp = -xp]; xp)
```

Test values: invnorm[0.001] outputs –3.0902323, and invnorm[0.75] outputs 0.67448975.

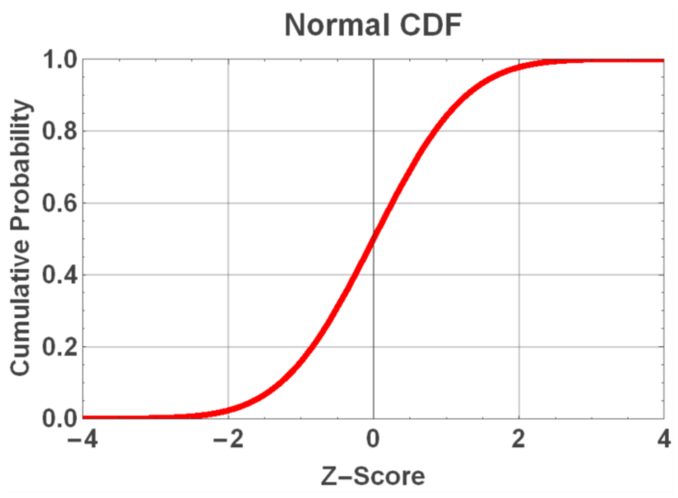


cdf(*z*)

The following Mathematica equation converts z-scores to probabilities (*p*):

```
cdf[z_] := (c1 = 2.506628275; c2 = .31938153; c3 = -.356563782; c4 = 1.781477937;
c5 = -1.821255978; c6 = 1.330274429; c7 = .2316419; (*program constants*)
If[z >= 0, w = 1, w = -1]; t = 1. + c7*w*z; y = 1./t;
cdfout = 0.5 + w (0.5 - (c2 + (c6 + c5*t + c4*t^2 + c3*t^3)/t^4)/(c1* Exp[.5 z^2] t)))
```

Test values: cdf[–1.98] outputs 0.023951694, and cdf[3.00] outputs 0.99865003.



binomialp

The following Mathematica program calculates the Binomial probability, *p* (X ≥ number of hits)

```
logComb[n_, k_] := Sum[Log[n - i + 1] - Log[i], {i, 1, k}]
binomialp[trials_, successes_] := Module[{p = 0.5, logP, logQ, pValue = 0}, logP = Log[p];
  logQ = Log[1 - p];
  For[k = successes, k <= trials, k++, logProb = logComb[trials, k] + k*logP + (trials - k)*logQ;
   pValue += Exp[logProb]];
  pValue]
```

Here are two test examples:

- For 16 trials with 6 hits, the Binomial *p* is 0.894943.
- For 14 trials with 12 hits, the Binomial *p* is 0.00646973.

This function calculates the total probability of observing a given number of successes or more, under the assumption of random success (absent mental influence) with a probability of 0.5 per trial. This probability is often used to determine the statistical significance of the observed success rate.

When Hit Rates for 1s and 0s differ

When a user’s hit rates for predicting 1s and 0s are not the same, denote HR_1 as the hit rate for predicting ‘1’ and HR_0 as the hit rate for predicting ‘0’. We can incorporate these differing hit rates into the Bayes’ updating process as follows:

If the observation (*obs*) is 1, calculate the total probability of the observation being a 1, *P*(*obs* = 1), and update the posterior probability:

1) Calculate the joint probability of the state being 1 and the observation being 1, which represents the probability of correctly predicting a 1:
   $P(1 = T \,\&\, obs = 1) = P(1 = T) * P(obs = 1|1 = T) = prior * \ HR_1$
2) Calculate the joint probability of the state being 0 and the observation being 1, which represents the probability of incorrectly predicting a 1:
   $P(1 = F \,\&\, obs = 1) = P(1 = F) * P(obs = 1|1 = F) = (1 - prior) * \ (1 - HR_0)$
3) Sum these two two probabilities to find the total probability of observing a 1:
   $P(obs = 1) = prior \ * HR_1 + (1 - prior) * \ (1 - HR_0)$
4) Use Bayes' Rule to update the posterior probability for $P(1 = T \mid obs = 1)$, which is the probability that '1' is the correct state or answer, given that a 1 was observed:
   $$P(1 = T \mid obs = 1) = \frac{HR_1 * \ prior}{HR_1 * \ prior + \ (1 - HR_0) * (1 - prior)}$$

If *obs* is 0, calculate the total probability of the observation being a 0, *P*(*obs* = 0), and update the posterior probability:

1) Calculate the joint probability of the state being 1 and the observation being 0, which represents the probability of incorrectly predicting a 0:
   $P(1 = T \,\&\, obs = 0) = P(1 = T) * P(obs = 0|1 = T) = prior * \ (1 - HR_1)$
2) Calculate the joint probability of the state being 0 and the observation being 0, which represents the probability of correctly predicting a 0:
   $P(1 = F \,\&\, obs = 0) = P(1 = F) * P(obs = 0|1 = F) = (1 - prior) * \ HR_0$
3) Sum these two probabilities to find the total probability of observing a 0:
   $P(obs = 0) = prior * \ (1 - HR_1) + \ (1 - prior) * HR_0$
4) Use Bayes' Rule to update the posterior probability for $P(1 = T \mid obs = 0)$, which is the probability that '1' is the correct state or answer, given that a 0 was observed:
   $$P(1 = T \mid obs = 0) = \frac{(1 - HR_1) * prior}{(1 - HR_1) * prior + \ HR_0 * (1 - prior)}$$

The following Mathematica program implements the solution found above to calculate the posterior taking as inputs, likelihood1 (*HR*$_1$, the user's *HR* for 1s), likelihood0 (*HR*$_0$, the user's *HR* for 0s), *prior* and *obs* (1 or 0):

```
(* Function to calculate the posterior probability *)
posterior[likelihood1_, likelihood0_, prior_, obs_] :=
  If[obs == 1,
    (* Case when obs=1 *)
    (likelihood1 * prior) / (prior * likelihood1 + (1 - prior) * (1 - likelihood0)),
    (* Case when obs=0 *)
    ((1 - likelihood1) * prior) / (prior * (1 - likelihood1) + (1 - prior) * likelihood0)
  ]
```

After each iteration, the calculated posterior probability will become the prior probability for the next update. Following each update, we identify the state (either ‘1’ or ‘0’) that has the highest probability of being correct based on the current posterior. Specifically, if the posterior probability is greater than 0.5, then ‘1’ is deemed the most probable state; if it’s less than 0.5, then ‘0’ is considered the most probable, and its probability is 1 – *posterior*.