Drift Diffusion Model Fundamentals and Python Analysis Workflow

Authors: Wang Qihui, Ren Ziwei, Hu Chuanpeng, Hu Chuanpeng

Date: 2025-10-14T00:00:00+00:00

Abstract

The Drift Diffusion Model (DDM) is a canonical computational model for studying human binary decision-making. This model assumes that individuals continuously sample and integrate evidence related to the options before making a choice, and that a decision is triggered and executed when the accumulated evidence reaches a preset decision boundary. The Hierarchical Drift Diffusion Model (HDDM) introduces a Bayesian hierarchical modeling approach on the basis of DDM. By simultaneously modeling parameters at both the individual and group levels, it enables information sharing across different hierarchical levels, thereby significantly improving the statistical power of parameter estimation. However, the appropriate use of HDDM requires researchers to possess corresponding theoretical foundations and technical capabilities, including understanding the model's applicable conditions, experimental design requirements, and mastering specific modeling and parameter inference methods. To help researchers apply HDDM in a standardized manner, this paper provides a systematic user guide. In the theoretical section, we start from the source theory of evidence accumulation models, provide a detailed introduction to DDM's model assumptions and its constraints on experimental design, to help researchers determine whether DDM is suitable for specific research questions. In the practical section, based on the dockerHDDM tool, we provide a complete standardized analytical workflow. Specifically, it includes standardized procedures such as Docker environment installation and dockerHDDM deployment, data preprocessing, model construction and fitting, model comparison, statistical inference, and results reporting. Finally, we also provide practical recommendations for the actual application of dockerHDDM. This paper aims to build a complete bridge from DDM theory to practical application, and to promote the standardized use of HDDM in cognitive research.

Full Text

Foundations of the Drift Diffusion Model and Its Python Analysis Workflow

Qi-Hui Wang; Zi-Wei Ren; Chuan-Peng Hu (School of Psychology, Nanjing Normal University; Corresponding author: Chuan-Peng Hu, E-mail: hcp4715@hotmail.com)

1. Overview

Cognitive modeling employs computational approaches to elucidate human cognitive processes. Among these, Evidence Accumulation Models (EAMs), also known as Sequential Sampling Models (SAMs), represent the predominant framework for modeling human decision time. These models posit that individuals continuously sample and integrate evidence relevant to choice options until the accumulated evidence reaches a predetermined decision boundary (threshold), at which point a decision is triggered and executed (Liu & Hu, 2024). On one hand, the theoretical assumptions of EAMs have received extensive support from neurobiological and behavioral experiments; on the other hand, EAMs have been widely applied across psychology, behavioral economics, neuroscience, and psychiatry to help researchers understand the cognitive processes underlying decision-making (Cavanagh et al., 2011; Herz et al., 2017; Shadlen & Shohamy, 2016).

The Drift Diffusion Model (DDM) is one of the most widely used computational models within the EAM framework (Ratcliff et al., 2016). Building upon EAMs, DDM further assumes that: (1) evidence accumulation is universal; (2) evidence accumulation is selective; (3) evidence accumulates linearly with noise; (4) decision criteria remain constant; and (5) the decision process is independent of response execution (Liu & Hu, 2024).

Specifically, the universality of evidence accumulation assumes that evidence can originate from any decision-relevant information, including sensory input, memory, economic value, or social value—whether from different sensory modalities or more abstract information (such as option values or memory-retrieved information). Through encoding and transformation, this information can be converted into abstract decision evidence for accumulation. The selectivity of evidence accumulation emphasizes that while decision-makers receive vast amounts of information during the decision process, only information relevant to the decision goal is encoded and transformed into evidence. The assumption of linear accumulation with random noise posits that evidence is integrated linearly while being subject to normally distributed random noise, thereby introducing stochastic variability into decisions. Through linear integration of sampled information, decision evidence exhibits a linear increase over time overall. The constancy of decision standards assumes that the criterion for forming decisions remains fixed across different decision processes. Once accumulated evidence reaches this stan-

dard, the decision-maker makes a decision. Thus, the decision standard does not change over time, and the required amount of evidence remains constant regardless of when the decision reaches the boundary. This assumption, combined with the third assumption, constitutes the "accumulation-to-boundary" pattern. The independence of decision and response execution distinguishes two processes: decision generation and response execution. DDM assumes these processes occur sequentially and independently. After a decision is formed, it must be executed through response processes such as key presses, eye movements, or verbal reports, which only begin after decision generation and do not influence the decision process itself (Liu & Hu, 2024).

As a widely used computational model, proper application of DDM requires researchers to deeply understand the decision-making process, particularly two aspects: first, the applicable conditions of DDM, which involve its theoretical assumptions and requirements for experimental design; and second, using DDM to fit experimental data, which involves employing specific software and tools to complete the modeling workflow and achieve reasonable parameter inference.

This chapter first introduces the theoretical foundations of DDM, including evidence accumulation theory and the derived Hierarchical Drift Diffusion Model (HDDM); then elaborates on HDDM's improvements over traditional DDM; next details Docker installation methods and dockerHDDM deployment procedures; subsequently introduces dockerHDDM's operation and standardized analysis workflow, including model construction, model fitting, model comparison (covering relative and absolute fit assessment), statistical inference, and results reporting; finally, we provide practical applications of dockerHDDM.

2. Theoretical Foundations

This section first introduces the core framework of the drift diffusion model—the evidence accumulation framework, then elaborates on its main parameters, and finally introduces its extended form—the hierarchical drift diffusion model, explaining its advantages over the standard model.

2.1 From Evidence Accumulation Framework to Drift Diffusion Model

Evidence accumulation theory (EAMs) assumes that when a stimulus is presented (e.g., an arrow pointing left or right), the decision-maker accumulates evidence for available actions or choice options (e.g., "Should I press the left or right arrow key?") until the evidence for one option reaches a threshold, immediately triggering motor execution of the overt response (e.g., pressing the left arrow key). Total reaction time (RT) is assumed to be the sum of three strictly sequential processing stages: (a) stimulus encoding, (b) decision process (evidence accumulation), and (c) motor response execution (Boag et al., 2025; Bompas et al., 2023; Kelly et al., 2021; Servant et al., 2021; Weindel, Gajdos, et al., 2021). Within the family of evidence accumulation models, models differ along several dimensions: whether evidence is discrete or continuous; whether

sampling occurs in continuous or discrete time; whether the amount of accumulated evidence is fixed or variable; and the type of decision stopping rule.

Based on different decision stopping rules, evidence accumulation theory has primarily developed two types of decision models: (1) absolute evidence accumulation models, which assume that people accumulate evidence for different options independently and simultaneously, thus applicable to any number of choice options; and (2) relative evidence accumulation models, which accumulate evidence based on the difference between two options.

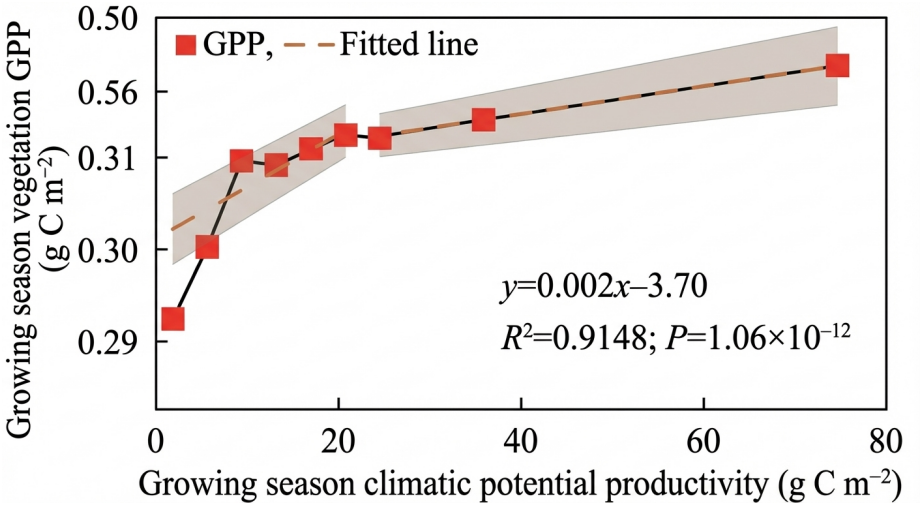

Figure 1: Figure 2

-1. Taxonomy of evidence accumulation models, adapted from Ratcliffe et al. (2016)

The typical computational model for relative evidence accumulation is DDM, which uses upper and lower boundaries corresponding to two different options and contains four basic parameters: v, a, z, and t. Drift rate (v) reflects the average speed of evidence accumulation toward one option; larger v leads to faster and more accurate decisions. As task difficulty increases and stimulus discriminability decreases, v becomes smaller. Decision boundary (a) represents the distance between the two thresholds; larger a requires more evidence, resulting in more cautious, slower, and more accurate responses. Initial bias (z) refers to the starting position of evidence accumulation, representing the initial bias (also called starting point). The closer the initial bias is to one boundary (1 or 0 corresponding to upper and lower boundaries, respectively), the faster and more frequent the corresponding response. Non-decision time (t) represents processing time unrelated to decision-making (such as sensory encoding and motor execution), which only shifts the RT distribution overall. DDM can also include three inter-trial variability parameters (s_v, s_z, s_t) to better

fit data. Drift rate variability (s_v) represents variability in drift rate across trials, increasing the proportion of slow errors. Starting point variability (s_z) refers to variability in initial bias across trials, increasing the proportion of fast errors. Non-decision time variability (s_t) represents variability in non-decision time across trials, simultaneously increasing the probability of extremely fast and extremely slow responses, making the RT distribution tails heavier. Since there is also normally distributed random noise with mean 0 in the evidence accumulation process, decision-making is essentially a stochastic process (Voss et al., 2013; Forstmann et al., 2016; Ratcliff et al., 2016).

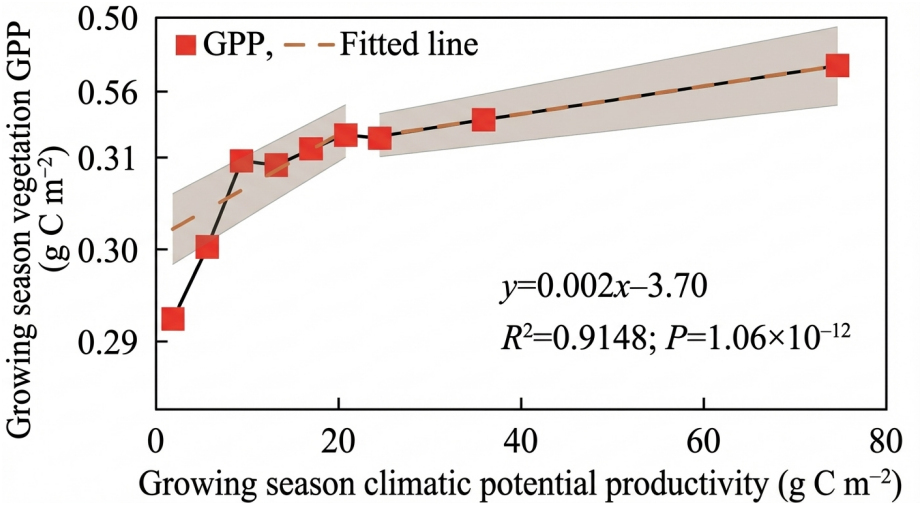

Figure 2: Figure 2

-2. Schematic illustration of the evidence accumulation process assumed by the Drift Diffusion Model (DDM)

2.2 Mainstream DDM Algorithms

Current drift diffusion models are primarily based on the Wiener process as the mathematical framework for modeling human decision processes. The Wiener First Passage Time (WFPT) is a fundamental concept in stochastic process theory, describing the time distribution characteristics of a Wiener process (standard Brownian motion) first reaching a specified boundary or threshold. This theory was first systematically developed by Klein (1952) and provides important mathematical foundations for modern decision modeling.

Mathematical definition and properties of WFPT: Consider a Wiener process $X(t) = x_0 + \mu t + \sigma W(t)$ starting at position x_0 with drift rate μ and variance σ^2, where $W(t)$ is standard Brownian motion. The first time this process reaches level a, $T = \inf\{t > 0 : X(t) = a\}$, follows an inverse Gaussian distribution with probability density function:

$$f(t) = \frac{|a - x_0|}{\sqrt{2\pi\sigma^2 t^3}} \exp\left(-\frac{(\mu t - (a - x_0))^2}{2\sigma^2 t}\right)$$

where $f(t)$ represents the probability density of first passage at time t, a is the target boundary position, and x_0 is the starting position. Therefore, $|a - x_0|$ represents the distance from the starting point to the target, corresponding to the decision boundary a in DDM. The term $\frac{1}{\sqrt{2\pi\sigma^2 t^3}}$ serves as a normalization factor, with σ^2 as the diffusion coefficient constraining noise intensity (random noise in the decision process). The t^3 term reflects the heavy-tailed nature of the distribution. The final 2π is a normalization constant. The exponential decay term $\exp\left(-\frac{(\mu t-(a-x_0))^2}{2\sigma^2 t}\right)$ controls the probability decay pattern over time. μ represents the average change speed of the stochastic process; when $\mu > 0$ and oriented toward the target, the Wiener process has a clear arrival tendency; when $\mu = 0$, the process becomes pure random walk with infinite mean arrival time, corresponding to the drift rate v parameter in DDM. μt is the expected drift distance within time t, while $(\mu t - (a - x_0))^2$ represents the squared deviation between the actual required distance and the expected drift distance.

2.3 Common DDM Software

The parameter estimation process for DDM involves complex multidimensional optimization and numerical calculations, placing high demands on researchers' mathematical modeling capabilities. As a classic computational model, several specialized analysis software packages have been developed to simplify DDM application. Currently, the most widely used tools include four main options: EZ-DDM, fast-DM, pyDDM, and HDDM. However, new toolkits continue to be developed, such as a seven-parameter DDM implementation in Stan (Henrich et al., 2023), PyBEAM (Murrow & Holmes, 2023), dynConfiR (Hellmann, 2024), and EMC2 (Stevenson et al., 2025).

These software packages employ different technical approaches and design philosophies, each with distinct characteristics. EZ-DDM provides fast closed-form solutions through mathematical simplification; fast-DM is based on the classic maximum likelihood estimation framework; pyDDM constructs a flexible Python ecosystem; and HDDM improves statistical power by building group-level parameter constraints on individual parameters. They exhibit different features and applicability across dimensions of algorithmic complexity, computational efficiency, statistical inference capability, and ease of use.

2.3.1 EZ-DDM EZ-DDM (EZ-Diffusion Model) is a simplified diffusion model estimation method developed by Wagenmakers et al., specifically designed for rapid analysis of two-choice reaction time experimental data (Wagenmakers et al., 2007). This method makes an important mathematical simplification that allows direct calculation of DDM's three core parameters. EZ-DDM converts observed behavioral data into cognitive parameters through

three simple mathematical transformation formulas. Input variables include mean reaction time (MRT), reaction time variance (VRT), and accuracy (P_c), while output parameters are drift rate (v), decision boundary (a), and non-decision time (t). The core feature of this method is providing closed-form solutions without requiring complex maximum likelihood estimation processes, enabling parameter estimation within seconds.

Its simplification is built upon three model assumptions: first, the starting point is at the midpoint between the two response boundaries, i.e., $z = a/2$, assuming participants have no prior preference for either response choice; second, no inter-trial parameter variability, setting $s_a = 0$, $s_z = 0$, $s_t = 0$, assuming cognitive parameters remain stable throughout the experiment, simplifying the complex parameter structure of the original diffusion model.

Due to these simplified constraints on DDM through model assumptions, EZ-DDM offers significant advantages in computational efficiency and usability. The method can quickly process large-scale datasets, completing parameter estimation within seconds, making it particularly suitable for studies requiring analysis of extensive datasets. It is also implemented in a dedicated R package, making it extremely convenient to use—researchers only need to provide basic summary statistics. Additionally, data requirements are relatively low, with stable parameter estimates obtainable from just 50-250 trials per condition.

However, the model assumption constraints of EZ-DDM also lead to its main limitations. First, the assumptions of boundary symmetry and no starting bias are frequently violated in real data, as many tasks exhibit natural response preferences. The stability requirement that parameters remain constant across trials ignores natural fluctuations in cognitive states such as attention, fatigue, and motivation. Second, there are limitations in task applicability—the method only works for single-decision-process tasks and cannot handle complex tasks requiring conflict resolution or multiple cognitive processes. Third, the tool exhibits estimation bias. When real data has substantial inter-trial variability, EZ-DDM systematically underestimates drift rate, with this bias amplifying as the true drift rate increases. Fourth, the method is sensitive to extreme accuracy values, with estimates becoming unstable when accuracy approaches 50% or 100%. Parameter estimation bias is more pronounced for high-ability individuals or simple experimental conditions. Fifth, validation is difficult—a notable limitation of EZ-DDM. Since it uses direct transformation rather than fitting optimization, the method always produces perfect data fit, creating a "perfect fit trap" that prevents assessment of model applicability through traditional goodness-of-fit metrics. Consequently, researchers must rely on additional model assumption tests to validate the method's suitability. Finally, EZ-DDM loses substantial information and cannot describe detailed RT distribution characteristics or error response time patterns.

Therefore, EZ-DDM is suitable as a screening tool and for exploratory analysis, or as an alternative when resources are limited. Before using EZ-DDM, researchers should first examine RT distribution right-skewness, differences be-

tween correct and error response speeds, and response preferences across different stimulus categories. When interpreting results, researchers should focus more on relative comparisons rather than absolute values, particularly exercising caution in interpreting potential underestimation effects under high drift rate conditions.

2.3.2 fast-DM fast-DM is an efficient DDM parameter estimation software developed by Voss and Voss, specifically addressing computational complexity and software usability issues in traditional diffusion model analysis (Voss & Voss, 2007). The software employs an innovative numerical solution method for partial differential equations (PDE), significantly improving parameter estimation speed and precision, making it one of the most widely used DDM analysis tools today.

fast-DM calculates the cumulative distribution function (CDF) based on PDE numerical methods, avoiding computational difficulties with infinite series convergence in traditional closed-form solutions. This method discretizes time and space dimensions, using numerical approximations to solve the Kolmogorov backward equation. The algorithm adopts the Kolmogorov-Smirnov statistic as the optimization criterion, estimating parameters by minimizing the maximum vertical distance between predicted and empirical distributions. The software uses an improved simplex downhill method for multidimensional parameter search, running three times consecutively to improve convergence precision. fast-DM also supports the complete Ratcliff diffusion model parameter set: decision boundary (a), starting point (z), drift rate (v), and three inter-trial variability parameters (s_z, s_v, s_t). Unlike EZ-DDM, fast-DM does not impose simplified constraints on the non-decision time (t) parameter and can handle complex asymmetric decision situations and inter-trial parameter fluctuations.

In terms of computational speed, the PDE-based method is faster than traditional approaches, enabling analysis of large sample datasets within reasonable timeframes. The software also allows balancing computational precision and speed through precision parameter adjustment. Additionally, fast-DM supports complex experimental designs, permitting certain parameters to vary across experimental conditions while others remain constant. The depends command enables flexible specification of parameter constraint relationships. The software provides rich control options, including fixing parameter values and setting initial values. It runs on both Linux and Windows platforms with user-friendly output formats convenient for subsequent statistical analysis, demonstrating strong usability and compatibility. Compared to EZ-DDM, fast-DM utilizes data more comprehensively, analyzing not just mean RT but complete RT distributions, including characteristics of correct and error response distributions. This comprehensive data utilization makes parameter estimation more precise and reliable.

However, greater freedom in assumptions brings steeper learning curves—the software requires users to possess certain DDM theoretical knowledge and

command-line operation experience. Control file syntax can be challenging for beginners, and the complexity of parameter settings may lead to analysis errors, along with longer computation times. Although significantly improved compared to traditional methods, computation time can still be substantial for high-precision requirements and large-sample analyses, with complex model parameter estimation potentially requiring several hours or more. Regarding inference results, fast-DM primarily focuses on point estimation, with relatively limited support for parameter uncertainty quantification and Bayesian analysis. It lacks commonly used confidence interval estimation and model comparison tools in modern statistical analysis, and because p-value interpretation of the Kolmogorov-Smirnov statistic differs from traditional statistical tests, users require deep cognitive psychology backgrounds for correct interpretation.

fast-DM's higher degree of freedom in assumptions indicates it is more suitable for research projects requiring precise DDM analysis, particularly when data quality is high and detailed modeling of inter-trial variability is needed. The software performs excellently in handling complex experimental designs and large-sample datasets. Therefore, fast-DM and EZ-DDM can be used in combination—for users needing rapid exploratory analysis or with limited DDM theoretical background, simpler tools (like EZ-DDM) can be used for preliminary analysis before employing fast-DM for refined modeling.

2.3.3 PyDDM PyDDM is specifically designed for implementing and fitting Generalized Drift Diffusion Models (GDDM). The software's core philosophy is to provide a highly flexible and extensible DDM modeling framework, enabling researchers to easily implement complex decision process models that traditional DDM cannot handle (Shinn, 2020).

PyDDM is based on the GDDM framework, calculating first-passage time distributions by solving the Fokker-Planck equation. The software implements three main numerical solution algorithms: forward Euler method, backward Euler method, and Crank-Nicolson method. These methods each have advantages, and the software automatically selects the most suitable algorithm based on model characteristics. PyDDM supports arbitrarily defined functions to describe DDM parameters. While traditional DDM assumes drift rate, boundary, and noise remain constant during experiments, PyDDM allows these parameters to depend on time, decision variable position, and task conditions. This flexibility enables simulation of complex mechanisms such as time-varying drift rates, collapsing boundaries, leaky integration, and position-dependent noise.

In computational efficiency, PyDDM can effectively utilize multi-core processors, enabling researchers to fit complex GDDM models with over 15 parameters within reasonable timeframes. PyDDM also includes a comprehensive model validation system, including parameter recovery tests and input-output checks, which are crucial for ensuring model result reliability, particularly when handling complex user-defined models. In addition to programming interfaces, PyDDM provides a graphical user interface allowing users to interactively explore

how model parameters affect RT distributions, valuable for both understanding model behavior and teaching demonstrations.

However, like fast-DM, excessive flexibility makes PyDDM require strong Python programming skills and deep DMM theoretical knowledge from users. The software's complexity may pose significant barriers for beginners. Customizing model components requires understanding object-oriented programming and numerical methods. Regarding computational speed, although significantly improved compared to trajectory simulation, fitting complex GDDM models still requires substantial computational resources. Computation time can become a limiting factor, particularly during parameter recovery tests or model comparisons. Finally, while PyDDM offers high flexibility, it may also lead to overly complex models. Therefore, users need to carefully balance model complexity and interpretability to avoid overfitting.

In summary, PyDDM is particularly suitable for methodological research, advanced modeling, and research projects requiring implementation of special DDM variants. The software excels in handling time-varying parameters, non-standard boundary conditions, or complex noise structures. For research requiring precise control over model details, PyDDM provides high flexibility.

2.4 Hierarchical Drift Diffusion Model

2.4.1 Advantages and Significance of Hierarchical Models HDDM (Hierarchical Drift Diffusion Model) is a DDM extension based on Bayesian hierarchical modeling. This software combines Bayesian statistical methods with hierarchical modeling techniques to specifically address the issue of insufficient statistical power in traditional DDM analysis (Wiecki et al., 2013).

For parameter estimation, traditional DDM analysis typically employs two extreme approaches: either assuming all subjects are completely independent (separate fitting) or assuming all subjects are identical (pooled fitting). HDDM provides a third solution through hierarchical modeling. In HDDM, each DDM parameter is modeled at two levels. Group-level parameters (e.g., μ_v and σ_v) describe distribution characteristics of the entire sample, while individual-level parameters (e.g., v_j) are drawn from the group distribution. Specifically, the system first estimates the drift rate distribution characteristics for the entire group (group mean μ_v and group standard deviation σ_v) from prior distributions, then each individual's drift rate is sampled from this group distribution.

This hierarchical structure achieves the statistical advantage of "information borrowing": individual parameters maintain some independence while being constrained by the group distribution. When between-subject similarity is high, the group distribution has smaller variance, imposing stronger constraints on individual estimates; when individual data are sufficient, the model still allows individual parameters to deviate substantially from group means. This avoids both estimation instability due to insufficient sample size in completely independent analyses and the problem of ignoring individual differences in pooled

analyses. Therefore, HDDM's advantage lies in its ability to utilize group-level information to improve individual parameter estimation accuracy, particularly when trial numbers are limited. Additionally, HDDM supports Bayesian inference, providing posterior distributions of parameters rather than just point estimates, enabling researchers to more comprehensively understand parameter uncertainty.

HDDM's greatest advantage is significantly improved statistical power. Simulation studies show that under the same sample size conditions, HDDM has higher effect detection capability than traditional methods. This advantage is particularly pronounced when trial numbers are small (20-50 trials per condition), with parameter recovery errors reduced by 20-50%. Meanwhile, based on the Bayesian framework, HDDM can provide complete posterior distributions rather than point estimates, allowing researchers to accurately quantify estimation uncertainty. This is valuable for both understanding parameter estimate reliability and conducting statistical inference. HDDM also supports complex experimental designs, including between-subject and within-subject factors and covariate analysis. The software's regression modeling function (HDDMRegressor) allows researchers to directly test how continuous variables such as brain activity (EEG or BOLD signals) affect DDM parameters, making it valuable for cognitive neuroscience research. For goodness-of-fit testing, HDDM provides multiple model comparison metrics, including Deviance Information Criterion (DIC) and Bayes factors, to help researchers select the most suitable model. Additionally, because extreme individual parameters are constrained by the group distribution, HDDM naturally has robustness against outliers.

2.4.2 Advantages of Bayesian Methods Bayesian methods allow researchers to infer complete posterior distributions for each parameter rather than just point estimates. This enables quantification of parameter estimation uncertainty, providing more comprehensive understanding of model parameters (Wiecki et al., 2013). HDDM employs hierarchical Bayesian models, allowing individual parameters to be sampled from group distributions. This structure not only utilizes between-subject similarities but also accounts for between-subject differences, thereby improving parameter estimation accuracy and efficiency. Consequently, with limited trial numbers, HDDM can provide more reliable parameter estimates through its hierarchical structure and Bayesian methods, which is particularly important in experimental designs with limited trials per condition.

HDDM using Bayesian sampling also supports complex model extensions, such as including inter-trial variability parameters (e.g., variability in drift rate and non-decision time) and regression models, enabling researchers to more flexibly handle various experimental designs and data analysis needs. Finally, Bayesian methods provide multiple model comparison and validation tools, such as DIC, Widely Applicable Information Criterion (WAIC), and Leave-One-Out Cross-Validation (LOO-CV), which help researchers select the most appropriate model

for the data and validate model fit.

3. DDM Model Assumptions and Experimental Design Constraints

Since the Hierarchical Drift Diffusion Model (HDDM) is based on standard DDM, it contains strict theoretical assumptions about the decision process, meaning the model is not suitable for all research scenarios. Similarly, the basic assumptions of DDM profoundly influence specific choices in experimental design. Based on guidelines from Boag et al. (2025), this section explains how to integrate experimental design with DDM modeling.

3.1 DDM Model Assumptions

Before using DDM analysis, it is necessary to determine whether DDM is appropriate for the current experimental task. Since DDM can output a series of latent cognitive process parameters representing group-level or individual-level processes (e.g., drift rate v, decision boundary a, initial bias z, and non-decision time t), DDM is suitable for research questions involving assessing whether model parameters differ within groups, between groups, or across experimental conditions, and how certain cognitive processes relate to individual-level covariates.

Research questions must also be mappable onto at least one parameter representing a latent process. For example, Ratcliff et al. (Ratcliff et al., 2003, 2004, 2006) tested a series of hypotheses: whether age-related response slowing originates from smaller drift rates (cognitive impairment hypothesis), decision boundaries (conservative response hypothesis), or longer non-decision times (physiological slowing hypothesis). This question clearly tests three competing hypotheses that can be instantiated and evaluated in DDM. Similarly, DDM can test whether relationships exist between individual-level covariates (such as EEG or BOLD signals) and certain parameters.

If research questions violate DDM's model assumptions—for example, investigating how drift rate changes over time in two-choice decisions—other extended models must be used, because DDM's basic assumption is that drift rate remains constant.

3.2 DDM Constraints on Experimental Design

After developing research questions suitable for DDM, the next step is to design experimental tasks that provide evidence for the research questions while satisfying DDM model assumptions. We now discuss DDM-specific constraints in task design, linking each constraint to relevant DDM model assumptions.

3.2.1 Single Response to Consistent Stimuli Since DDM assumes decision-making involves a single, uninterrupted evidence accumulation stage

that ultimately produces a discrete two-choice response, selected tasks must have clearly defined stimulus onset and response completion times that do not overlap with processes outside the response window. DDM also assumes that during decision-making, the evidence accumulation process has consistent and stationary input, requiring experimental tasks to ensure two points: first, experimental stimuli have fixed "intensity" within trials and preferably remain presented throughout the response window (from stimulus onset to response) to ensure continuous evidence input during decision-making; second, even if stimuli are briefly flashed (e.g., in visual signal detection paradigms), it must be assumed that their persistent representation can be maintained in visual short-term memory long enough to support decision-making (Ratcliff & Rouder, 2000; Smith & Ratcliff, 2009), ensuring consistent evidence input throughout the accumulation process.

Furthermore, each decision should end with a single, discrete response from two options. This is because in DDM, decisions always terminate when evidence reaches a single response threshold (upper/lower boundary). Therefore, tasks involving open-ended response options (e.g., free recall tasks) or requiring multiple responses within one trial (e.g., attentional blink tasks) require DDM with additional extensions.

3.2.2 DDM Constraints on Within-Trial Stationarity DDM assumes that model parameter magnitudes do not systematically change during the decision process. That is, threshold and initial bias magnitudes do not change due to stimulus characteristics, and drift rate remains constant from stimulus presentation until decision-making. Therefore, any information intended to influence threshold or initial bias must be presented before stimulus onset. Evidence input to the decision process should also not change systematically within a single trial—stimulus features relevant to decision-making (such as representations in visual short-term memory) should remain stable throughout the trial (Smith & Lilburn, 2020). For example, in perceptual decision tasks, stimulus brightness or contrast should not change mid-trial, otherwise drift rate would need adjustment. For tasks involving dynamic evidence, other more suitable models are recommended, such as those proposed by Diederich (2024), Diederich & Trueblood (2018), Holmes et al. (2016), and Holmes & Trueblood (2018).

3.2.3 DDM Constraints on Within-Condition Stationarity DDM assumes that stimulus presentation remains constant across trials within the same condition because the model requires treating trials from the same condition as independent observations of the same underlying cognitive process. Except for non-systematic inter-trial variability explained by cross-trial variability parameters in the model (e.g., $s_t/s_v/s_z$), thresholds or drift rates should not systematically vary across trials within the same experimental condition. This assumption is important for statistical power and measurement precision, which depend on convergent information across trials (Smith & Little, 2018). When designing experiments, researchers should avoid factors that cause systematic

parameter changes across trials. For example, drift rate is known to increase with learning, initially rising sharply then gradually stabilizing (e.g., Fontanesi et al., 2019; Miletić et al., 2021; Pedersen et al., 2017; Sewell et al., 2019). Drift rate may also decrease due to fatigue or inattention (Huang-Pollock et al., 2020; Ratcliff & Van Dongen, 2011; Walsh et al., 2017), and thresholds may gradually decrease as participants become impatient during experiments (Hawkins et al., 2012; Larson & Hawkins, 2023).

However, decision processes contain noise at many different levels, including inherent noise in the nervous system (Faisal et al., 2008; P. L. Smith, 2010, 2023) and dynamic fluctuations in cognitive and affective states (Miletić et al., 2024; Schurr et al., 2024), making inter-trial variability unavoidable (Aschenbrenner et al., 2018; Rouder et al., 2023). Standard DDM explains such noise sources through cross-trial variability parameters (e.g., $s_t/s_v/s_z$). Nevertheless, researchers should take reasonable measures to ensure such variability remains as non-systematic as possible.

3.2.4 DDM Constraints on Stimulus Settings Stimuli provide the evidence upon which decision processes are based and largely determine the cognitive domain involved in the task. For example, in psychophysical tasks, evidence may be based on objective stimulus brightness values (e.g., Sewell & Smith, 2012; van Ravenzwaaij et al., 2020). In contrast, in preference choice tasks, evidence may be subjective values evoked by viewing food images (e.g., Huseynov & Palma, 2021; Milosavljevic et al., 2010). In working memory and categorization tasks, evidence may originate from activation strength of items in memory (Ratcliff, 1978; Shadlen & Shohamy, 2016) or strength of learned associations between stimuli and expected response outcomes (Dutilh et al., 2009; Dutilh, Krypotos, & Wagenmakers, 2011; Miletić et al., 2021; Sewell et al., 2019). As previously mentioned, evidence provided by stimuli should be fixed within a trial (i.e., constant intensity during the trial) to provide consistent (stationary) input for the decision process.

Since different stimulus levels typically aim to affect the signal-to-noise ratio of evidence in the decision process (e.g., discriminability, difficulty), it is important to calibrate stimuli to appropriate difficulty levels when designing experiments because DDM may have difficulty fitting floor and ceiling effects (Dutilh, Wagenmakers, et al., 2011). Floor effects occur when tasks are too difficult, typically meaning participants cannot distinguish choice options, so participants may respond by guessing rather than sampling evidence as DDM assumes. In contrast, ceiling effects occur when tasks are too simple, resulting in very few error observations. Sufficient error observations are important for reliable model estimation (Lüken et al., 2025). Therefore, it is recommended to calibrate stimuli to produce error rates between 5% and 35% (Dutilh, Wagenmakers, et al., 2011; Lüken et al., 2025; Ratcliff & Childers, 2015).

3.2.5 DDM Constraints on Response Methods DDM assumes that response onset coincides with the termination time of the evidence accumulation process. Decision and motor response processes occur sequentially; therefore, response methods that allow precise RT measurement (such as manual key presses or saccades) should be considered to avoid reducing estimation precision of non-decision time.

3.2.6 Mapping Experimental Manipulations to DDM Parameters Establishing associations between experimental manipulations (e.g., speed-accuracy instructions, task difficulty, or working memory load) and DDM parameters is important. Not all DDM parameters are relevant to every analysis. For example, researchers studying consumer choice preferences (e.g., preference for one product over another) may not be interested in non-decision time but may be very interested in using drift rate to measure preference strength and starting point (or threshold) to measure choice bias (Busemeyer & Townsend, 1993; Cerracchio et al., 2023). Additionally, it is common practice not to estimate variability parameters (e.g., $s_t/s_v/s_z$, by fixing them to zero) unless they are needed to explain certain data features (e.g., fast guessing; Lerche & Voss, 2016; Ratcliff & Rouder, 1998).

4. Implementing Hierarchical Bayesian DDM Analysis with dockerHDDM

As a mature Python library, HDDM contains many useful built-in functions. Recently, Pan et al. (2025) developed dockerHDDM, which further simplifies installation and achieves cross-hardware compatibility, making it an excellent tool for DDM analysis. This section is divided into two parts: first, introducing usage methods and parameter settings for existing functions in HDDM; then explaining dockerHDDM's advantages in installation, compatibility, and functionality. This helps readers understand HDDM parameters while gaining deep knowledge of dockerHDDM's operating environment and logical framework.

4.1.1 Reaction Data Encoding

In HDDM, different response encoding modes actually correspond to different model fitting approaches; therefore, we first introduce how to encode response data. In the data table (see Section 5.1 Data Preprocessing), there are two optional encoding methods for the response column: (1) accuracy encoding; (2) stimulus encoding.

For accuracy encoding, use 1 for "correct" responses and 0 for "incorrect" responses in the response column. HDDM treats correct and incorrect responses as lower and upper boundary responses, respectively. Under this encoding, drift rate v reflects the accumulation speed of effective evidence—higher drift rate leads to faster acquisition of evidence for correct decisions. Since subjects have no prior preference for correct or incorrect responses when making deci-

sions, the starting point parameter z is generally fixed when using accuracy encoding (excluding z from the include parameter, in which case $z = 0.5$, i.e., the midpoint or one-half of the decision boundary).

For stimulus encoding, 1 and 0 correspond to two different choices (e.g., encoding leftward arrows as 1 and rightward arrows as 0). This encoding method is particularly suitable when assuming subjects have response preferences (e.g., subjects may be more inclined to believe arrows point left). Under this encoding, drift rate reflects the accumulation speed of stimulus or decision feature information—higher drift rate leads to faster acquisition of specific information (e.g., leftward stimulus). If individuals have processing or response biases toward "leftward stimulus features," the starting point may be biased toward the corresponding decision boundary.

4.1.2 Basic Models

To create a basic hierarchical drift diffusion model, use the following code to define the model:

```
model = hddm.HDDM(data, include = (), depends_on = {}, p_outlier
= 0.05, informative = True)
```

-1. Basic model function parameter introduction

Function/Parameter	Description
hddm.HDDM	Function name for creating a hierarchical Bayesian drift diffusion model (HDDM) object. Its internal function parameters determine specific model settings.
data	Function parameter name representing the input data frame, which must contain 'rt' (reaction time), 'response' (response encoding), and 'subj_idx' (subject identifier) columns.
include	Function parameter name specifying DDM parameters to include in the model for extending the basic DDM structure. Optional values: 'z' (starting point bias), 'sv' (drift rate inter-trial variability), 'st' (non-decision time inter-trial variability), 'sz' (starting point inter-trial variability).

Function/Parameter	Description
depends_on	Function parameter name specifying how model parameters depend on experimental conditions. Keys are parameter names (e.g., 'v', 'a'), values are column names in the data indicating that the parameter varies across levels of this column.
p_outlier	Function parameter name representing the prior probability of outlier trials in the data frame, used to handle values not conforming to DDM.
informative	Function parameter name indicating whether to use informative prior distributions for group-level standard deviation parameters, default True (use informative priors).

The basic model is suitable for accuracy-encoded data, where model is the custom name for the fitted model, data corresponds to the dataset used for model fitting, and the include parameter determines which parameters the fitted model contains. More included parameters result in slower model fitting and greater resource consumption (especially for inter-trial parameters 'sv', 'st', 'sz'). See Section 6 for discussion on when to fit inter-trial parameters.

You can also fit different parameter values for different experimental conditions by setting the depends_on parameter, for example: `model = hddm.HDDM(data, depends_on={'a': 'drug', 'v': ['drug', 'difficulty']})` will fit different thresholds (a) for different drugs and independent drift rates (v) for all drug-difficulty combinations. In practice, you can freely define the depends_on parameter based on experimental purposes to flexibly specify arbitrarily complex models. Additionally, even if trials with excessively long or short RTs have been removed during data preprocessing, it is best to set `p_outlier = 0.05` in the model. This is equivalent to assuming about 5% of trials may be outliers that the model cannot explain. This parameter provides a flexible "buffer zone" for the model, preventing overfitting to these noisy trials and enabling more accurate capture of core cognitive parameters in the decision process. Since HDDM sets well-validated informative priors that appropriately constrain reasonable parameter ranges, it is generally recommended to keep `informative=True` as the default setting unless you have extremely large amounts of data or very clear reasons for using non-informative priors. However, note that because the depends_on parameter independently calculates corresponding parameters for different conditions, ignoring within-subject parameter correlations, it is better to use regression (regressor) models when fitting within-subject experimental

design data.

4.1.3 Regression Models

Like basic models, regression models default to using accuracy encoding (regression models using stimulus encoding will be discussed in Section 3.1.5). We can build regression models using:

```
model = HDDRegressor(data, 'regression_formula', include = (),
depends_on = {}, p_outlier = 0.05, group_only_regressors = True,
keep_regressor_trace = True, informative = True)
```

-2. Regression model function parameter introduction

Function/Parameter	Description
HDDRegressor	Function name for creating a hierarchical regression Bayesian drift diffusion model object. Its internal function parameters determine regression model details.
data	Function parameter name, same as hddm.HDDM, representing the input data frame, which must contain 'rt' (reaction time), 'response' (response encoding), and 'subj_idx' (subject identifier) columns.
include	Function parameter name, same as hddm.HDDM, specifying DDM parameters to include for extending the basic DDM structure. Optional values: 'z' (starting point bias), 'sv' (drift rate inter-trial variability), 'st' (non-decision time inter-trial variability), 'sz' (starting point inter-trial variability).
p_outlier	Function parameter name, same as hddm.HDDM, representing prior probability of outlier trials in the data frame for handling values not conforming to DDM.

Function/Parameter	Description
group_only_regressors	Function parameter name determining whether to evaluate parameters only at the group level or also estimate for each subject, default True. When set to False, random slopes are simultaneously set for parameters, assuming between-condition changes for each subject follow a normal distribution (like random slopes in linear mixed models).
keep_regressor_trace	Function parameter name indicating whether to save regression coefficient traces in MCMC sampling, default True. If set to True, the MCMC sampler saves each regression coefficient's value at every iteration, enabling not only point estimates but also complete posterior distributions.
informative	Function parameter name, same as hddm.HDDM, indicating whether to use informative prior distributions for group-level standard deviation parameters, default True (use informative priors).

HDDM's regression formula syntax is the same as in R, for example: `a ~ 1 + C(drug, Treatment('control')), v ~ 1 + C(drug, Treatment('control'))*C(difficulty, Treatment('Low'))`. Here, `~ 1` indicates building a regression model with intercept (0 means no intercept), using the level inside `Treatment()` as the intercept term (the drug column must contain this level). For `v ~ 1 + C(drug, Treatment('control'))*C(difficulty, Treatment('Low'))`, `*` indicates considering both main effects and interaction terms, while `:` connecting two conditions considers only interaction effects. `group_only_regressors` indicates whether to fit parameters only at the group level. When setting `group_only_regressors = False`, random slopes are simultaneously set for parameters, assuming between-condition changes for each subject follow a normal distribution (like random slopes in linear mixed models). Therefore, setting this parameter to False can improve model fit but consumes more resources and time. `keep_regressor_trace` directly affects whether regression coefficient traces are saved. If set to True, the MCMC sampler saves each regression coefficient's value at every iteration, enabling not

only point estimates but also complete posterior distributions. After obtaining complete posterior distributions, you can not only perform convergence diagnostics by plotting regression coefficient trace plots but also better visualize parameter differences across conditions. Regression models can also estimate trial-by-trial covariates' (e.g., brain metrics fMRI) immediate effects on DDM parameters. If you predict that activity in a certain brain region has a linear relationship with drift rate, you can set the regression formula as follows (ensure the data frame contains a column recording that brain region's activity, here assumed to be named BOLD): `m = hddm.models.HDDMRegressor(data, 'v ~ BOLD:C(trial_type)')`. HDDMRegression automatically adds intercept and slope for BOLD's trial-by-trial measurements and estimates this set of parameters separately for each level of `trial_type`. `C(trial_type)` tells the model to treat `trial_type` as a categorical variable and handle it with dummy coding.

4.1.4 Stimulus Coding Models

In certain experimental situations, we need to consider starting point effects or handle choice tasks without clear right/wrong distinctions (e.g., "Do you prefer bananas or potato chips?"). For such cases, stimulus coding models can be used:

```
model = hddm.HDDMStimCoding(data, include = (), depends_on = {}, stim_col='stim', split_param='v', p_outlier = 0.05, informative = True)
```

-3. Stimulus coding model function parameter introduction

Function/Parameter	Description
hddm.HDDMStimCoding	Function name for creating a stimulus-coded hierarchical Bayesian drift diffusion model.
data	Function parameter name, same as hddm.HDDM, representing the input data frame, which must contain 'rt' (reaction time), 'response' (response encoding), and 'subj_idx' (subject identifier) columns.
include	Function parameter name, same as hddm.HDDM, specifying DDM parameters to include for extending the basic DDM structure. Optional values: 'z' (starting point bias), 'sv' (drift rate inter-trial variability), 'st' (non-decision time inter-trial variability), 'sz' (starting point inter-trial variability).

Function/Parameter	Description
depends_on	Function parameter name, same as hddm.HDDM, specifying how model parameters depend on experimental conditions.
stim_col	Function parameter name specifying the column name in the data frame representing stimulus types (e.g., 'left' vs 'right').
split_param	Function parameter name specifying which parameter to split based on stimulus type, usually 'v' (drift rate).
p_outlier	Function parameter name, same as hddm.HDDM, representing prior probability of outlier trials.
informative	Function parameter name, same as hddm.HDDM, indicating whether to use informative prior distributions.

Figures

Figure 3: Figure 3

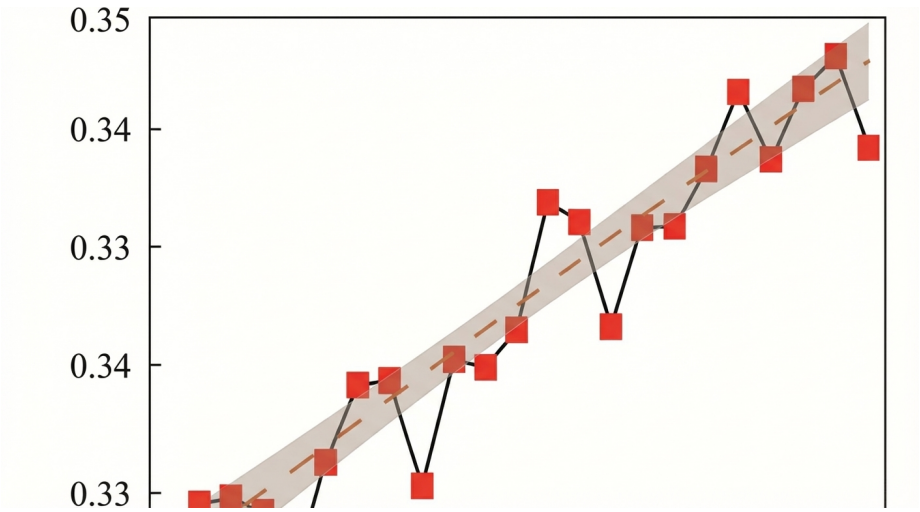

Figure 4: Figure 4

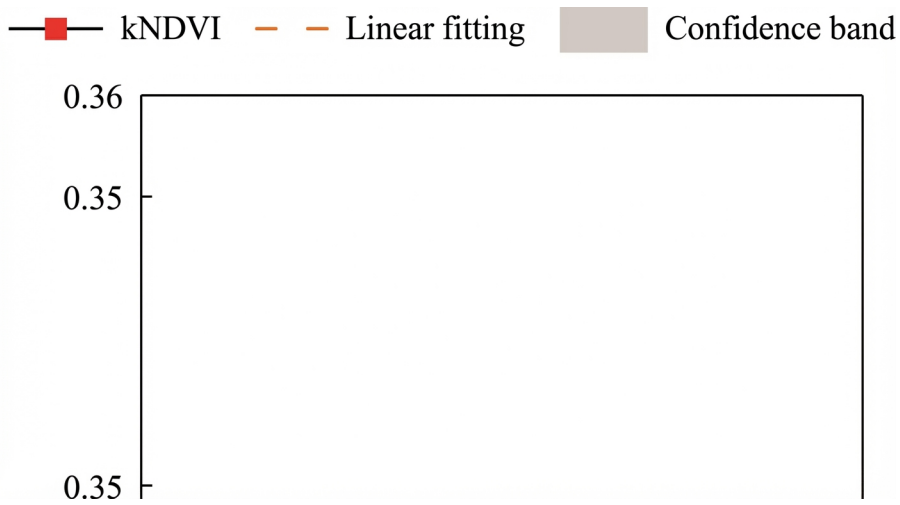

Figure 5: Figure 6