Interpreting and Steering LLM Agents for Social Simulations

Jiayue Gaveal Fan
UC Berkeley
gaveal.fan@berkeley.edu

Arul Murugan
UC Berkeley
arul@berkeley.edu

Shreyas Krishnan
UC Berkeley
shreyas.krishnan@berkeley.edu

Abhishek Nagaraj
UC Berkeley
nagaraj@berkeley.edu

January 11, 2026

Abstract

Generative agents built on large language models (LLMs) have proven to be powerful for simulating human behavior, making them valuable additions to the social scientific toolkit. However, the black-box nature of LLMs poses a major roadblock to the two factors social scientists demand of experiments: (i) *interpretability*: explaining what internal factors drive agent behavior; and (ii) *controllability*: reliably shifting behaviors along theoretically meaningful dimensions. We demonstrate how the black box could be opened up by interpreting and controlling two foundational components of human behaviors, namely preferences (risk attitudes, altruism) and capabilities (divergent creativity, product innovation), using four classic economic and creative tasks implemented as natural-language interactions. We examine three types of control methods: (1) prompt-based persona manipulation, (2) sparse autoencoder (SAE) for mechanistic interpretability and feature-based steering, and (3) probe for representational interpretability and direction-based steering. Overall, our results show that emerging interpretability techniques can often be superior to prompt-based methods of control. SAEs prove to be strong tool for mechanistic understanding and steering of complex behaviors, and probes offer an efficient option for controlling simpler and pre-defined traits. We discuss implications of these methods for future work using LLM agents for social scientific simulations.

1 Introduction

Generative agents produced by Large Language Models (LLMs) mimic humans in multiple dimensions: they understand natural languages, can communicate with humans, and have similar behavioral patterns with humans (Anthis et al. 2025). They can be configured to represent wide-ranging background profiles and respond to different experimental scenarios at a lower cost than observing the behaviors of an equivalent number of humans (Park et al. 2024). Given these advantages, social scientists have been motivated to use generative agents to conduct social simulations (Horton 2023, Chen et al. 2023). Further, generative agents can also be trained to mimic different personas of real humans, which allows social scientists to study heterogeneity in human behavior. With generative agents based simulations, social scientists can develop new theories of human behavior more cost-effectively and more efficiently by conducting experiments with these agents rather than relying solely on human participants (Tranchero et al. 2024).

Despite their advantages, the black-box nature of generative agents has been proven to be a major roadblock for social scientific simulations (Anthis et al. 2025). Social scientists build new theories either inductively, by uncovering the mechanisms underlying observed behaviors, or deductively, by deriving hypotheses from abstract concepts and testing them through quantitative or experimental methods (Davis and Bingham 2007, Hedström 2005). Black-box LLMs don't allow for either of these approaches. It is difficult to identify the specific mechanisms driving behaviors for induction, and equally challenging to steer those mechanisms in a controlled manner for deduction.

Here, we offer some practical guidance that helps computational social scientists and economists open the black box of generative agents in simulations with the aim to bring back greater experimental control. We compare the effectiveness of standard prompt-based methods of control (where LLM agents are prompted to assume certain personas) with emerging techniques from the CS literature on mechanistic interpretability (Nanda 2023). Specifically, we examine the effectiveness of prompting as compared to sparse autoencoders (SAEs), and probes for understanding and steering agent behavior. Sparse autoencoders are specially trained neural networks that learn to decompose model activations into interpretable features, while probes are simple classifiers trained to detect whether specific concepts are encoded in a model's internal representations.

We focus on simulating two key dimensions of human behavior, specifically preferences and capabilities, across a set of classic experimental tasks. For preferences, we look at the lottery game (risk) (Edwards 1954) and the ultimatum game (altruism) (Camerer and Thaler 1995). For capabilities, we examine simple tasks capturing divergent creativity and product innovation. For each game, we use SAEs to understand what features are activated within the model with a view to uncovering mechanisms potentially driving agentic behavior. We then build probes for the central feature driving model behavior (for e.g. risk-taking intention in the case of lotteries) and compare the effectiveness of using probe- and SAE-based method of steering these features (e.g. amplifying the desire for risk-taking) with traditional prompt based methods of steering.

Overall we find that these techniques perform more favorably as compared to prompt-based methods of control. Specifically, (1) SAE-based methods provide both better understanding and greater control of agent behavior as compared to prompt-based methods; (2) Probe-based methods exhibit greater steerability on pre-defined traits of the agents while SAE-based methods are capable of exploring and steering previously unknown traits. Our approach illustrates how social scientists can gain deeper insight into and exert finer control over generative agent simulations by leveraging the representational and mechanistic understanding of agent behavior.

We make three contributions to the emerging literature of AI simulations in social science (Horton 2023, Park et al. 2024, Kazinnik 2023). First, we provide the first exploration of prompt-based, mechanistic interpretability (SAEs), and representational interpretability (probes) approaches for understanding and steering LLM agent behavior in social science simulations. Second, we demonstrate how SAEs and probes can provide interpretable, fine-grained control over agent behaviors along major human-like dimensions, surpassing the capabilities of prompt-based techniques alone. And finally, we validate our framework across a set of classic experimental tasks that measure both preferences and capabilities, providing early proof-of-concept for leveraging mechanistic and representational interpretability in generative agent simulation. We also contribute to the emerging literature on mechanistic interpretability

2 Related Work

2.1 Using Generative Agents in Computational Social Science

Social scientists have long leveraged the power of digital transformation to process massive data and address social scientific questions (King 2011). Recently, the adoption of machine learning and artificial intelligence as a methodological approach to both inductive and deductive (Davis and Bingham 2007) research has aided computational social scientists in a comprehensive scale. More recently, the adoption of machine learning and artificial intelligence as methodological approaches to both inductive and deductive research (Davis and Bingham 2007) has aided computational social scientists at scale. For example, text mining via machine learning tools aids pattern recognition for inductive research (Lazer et al. 2009, Salganik 2017). On the other hand for deductive research, advancements in computational capability have enabled researchers to use methods such as agent-based modeling to deduce macro-level patterns from sets of micro-level rules (Epstein 1999, Macy and Willer 2002).

Given the recent rapid development of LLMs out of traditional computational methods, we ask whether LLMs can serve as a new methodological instrument for the social sciences. Current research presents LLMs as a tool that leverages artificial intelligence systems designed to understand and generate human language by learning from large-scale human-produced text data (Llama Team 2024). With language as its interface, LLMs are capable of communicating with others, including humans and other LLMs. This allows LLMs to be capable of simulating human behaviors through their output texts. Their similarity with humans happens in both individual and group settings. LLMs are able to generate similarities of real-human level of heterogeneity in demographics (Park et al. 2024), economic rationality (Chen et al. 2023, Xie et al. 2025), cognitive abilities (Toubia et al. 2025), and other fundamental attributes (Abdulhai et al. 2024). LLMs also carry social abilities that allow them to organize social events in artificial settings (Park et al. 2023), and can generate networks whose structures are similar to human social networks (Chang et al. 2024). The outstanding performance of LLMs has prompted researchers to replicate experiments traditionally conducted on humans using generative agents powered by LLMs (Tranchero et al. 2024).

Using generative agents in experiments offers several advantages, including low cost, easy accessibility, and a close resemblance to real-human behaviors. Complementing the deterministic nature of agents in traditional agent-based modeling, generative agents introduce real-world contextual nuance and variability into simulations (Gao et al. 2024). Hence, generative agents are helpful

in multiple avenues: piloting untested experiments to observe possible outcomes and improving existing deductive tools, which complement the traditional data-driven approach in using machine learning to social scientific research (Grimmer et al. 2021).

Despite their many benefits, generative agents behave like a black box - we cannot clearly interpret why they act as they do, nor reliably steer them toward specific mechanisms (Jackson et al. 2025, Ludwig and Mullainathan 2024). The interpretability problem arises because LLMs are trained on massive, weakly documented corpora, making it impossible to know whose voices and assumptions are represented; this turns them into "stochastic parrots" with little accountability for output quality (Bender et al. 2021). Even when post-hoc explanations are available, they rarely reflect the model's true reasoning and can mislead downstream analyses (Rudin 2019). The controllability problem follows from this opacity: the same observed behavior can emerge from entirely different internal mechanisms, meaning researchers cannot isolate or manipulate the causal processes they wish to study. As a result, generative agents risk obscuring rather than clarifying the mechanisms behind theories. However, if we can open this black box, we gain the ability to both understand the mechanisms driving the agents' behaviors and intentionally modify them, expanding the credible uses of generative agents for the theory building in computational social science.

2.2 Opening the Black Box

Most LLMs are built on the Transformer architecture, which represents texts as numerical embeddings and processes them through multiple layers of attention and feedforward networks to predict probabilities of subsequent tokens (Vaswani et al. 2017). At the core of these models are weights, which are the parameters learned during training. Usually, there are billions or trillions of parameters trained for one model and they determine how strongly different parts of the input influence each other. When texts are fed into the model, they are transformed into embeddings, producing activations that encode linguistic, semantic, and contextual information. "Attention weights" in the Transformer architecture are specifically assigned to understand contextually relevant information, allowing LLMs to capture complex dependencies, such as syntax or even abstract reasoning (Vaswani et al. 2017).

Given the way LLMs are constructed, one approach to opening the black box is to prompt them to reveal their intermediate reasoning or "chain of thought" in their output. This approach demonstrates the behavioral codes of generative agents via natural language, making text-based analysis of their underlying reasoning processes possible (Xie et al. 2025). It also opens the black box as output generates, making the reasoning process accessible as the behavioral patterns. However, prompting generative agents about their reasoning also has three main drawbacks: First, what they say is not what they do. Unfaithful explanations always happen in chain-of-thought prompting as the generation process of explanations also suffers from structural biased information input from the model, disabling the agents to always give correct reasoning (Turpin et al. 2023). Second, for the same reason as the first drawback, generative agents might omit information in the chain of thought when prompted to reason. Third, researchers need multiple rounds of prompt engineering to get the granularity wanted for specific behavioral codes. Achieving a sweet spot between too abstract behavioral principles and too detailed reasons based on contextual information is difficult especially when the researchers want to observe multiple factors behind one behavioral pattern. Furthermore, chain-of-thought is not suitable for steering agent behaviors, as modifying generated explanations does not reliably translate into changes in the underlying decision process of

agents.

Another approach to opening the black box is to investigate the interpretability of the models, which could be divided into two focuses: representational interpretability and mechanistic interpretability (Alain and Bengio 2017, Elhage et al. 2021). Probing, trained on labeled datasets and predefined concepts, is useful to reveal the hidden representations of a neural network, hence showing what information can be decoded from them (Alain and Bengio 2017, Hewitt and Manning 2019, Belinkov 2022). When applied to decode the behavioral reasoning of generative agents, probing can serve as a measure of specific behavioral dimensions, such as risk, to assess the extent to which agents' actions reflect that characteristic. Representational interpretability is the most useful if the behavioral code is known and the researchers have labeled datasets to train a probe on that trait. Mechanistic interpretability is different, as it opens the black box by reverse engineering the computations within the transformer layers, for understanding which activations are responsible for what behaviors (Elhage et al. 2021, Nanda 2023). SAEs are the tools to achieve mechanistic interpretability, as they are small neural networks trained to reconstruct a model's internal activations using a sparse latent code (Bricken et al. 2023). Because of sparsity, the latent represents a relatively distinct pattern, and its dimensions act as interpretable "features". Contextualized in the generative agents' behaviors, SAEs can identify the top-activating features within the model that are responsible for their corresponding behaviors. This adds more explainability and causality to simulations using generative agents.

3 Methodology

It is to the best of our knowledge that none of the existing methods for opening the black box of generative agents in social science simulations have been systematically explored or compared. Previous attempts have utilized SAEs to demonstrate the steerability of generative agents in financial activities, as one example of social capabilities (Chen et al. 2025). We would like to push this methodological advance further by proposing to use prompting, SAE-based, and probe-based interpretability tools to understand generative agents' behaviors during simulations and compare their usefulness in interpretability and controllability.

In this paper, we demonstrate our ways of opening the black box by focusing on individual agents' traits, as they are the foundations of other levels of analysis in social sciences, such as groups, teams, organizations, and societies. We showcase two fundamental components of generative agents that mimic human behaviors: Preferences and Capabilities. We selected social experiments that have been conducted on humans and were proven certain traits were involved in the decision-making processes of experiment subjects. Existing results from these experiments help us interpret the mechanisms demonstrated by generative agents and verify if their behaviors are coherent with real humans if we prompt or steer the agents to have a specific characteristics.

Table 1 lists the tasks and the settings for our simulations with generative agents. We begin with the lottery (Edwards 1954) and the ultimatum game (Camerer and Thaler 1995) paradigms, which are widely used in psychology to study risk and altruism as core behavioral tendencies in one's preferences. We ask 40 agents to perform the task in 3 scenarios: baseline, where nothing is changed to alter the agents' behaviors; prompting, where we use prompts to change the agents' persona to be more risk-taking; and steering, where we find the feature activation for risk-taking behaviors and manually steer it to a high value. Similarly, we find divergent creativity and product innovation as

	Task Description	Settings
Preference		
Risk	Lottery: Focal agents are asked to choose between two options: 1. (Safe) Guaranteed 50 tokens; 2. (Risky) 50% chance of n tokens, 50% of 0 tokens	40 agents, 3 scenarios, $n = [10, 180]$
Altruism	Ultimatum Game: Focal agents are asked to accept or reject the offer: the proposer was given 100 tokens and has decided to offer you n tokens out of their 100 tokens. If accept, you keep the proposed amount; if you reject, both of you get 0.	40 agents, 3 scenarios, $n = [10, 100]$
Capability		
Divergent Creativity	List very detailed ways you can use a brick.	4 scenarios
Product Innovation	List as many specific enhancements as you can that would make a stapler better.	4 scenarios

Table 1: Implemented Experiments

important components of capabilities and use tasks similar to Torrance Tests of Creative Thinking (Torrance 1966) to see how prompting and steering can change the generative agents' performances. In addition to baseline, prompting, and steering, we added one scenario to the capability-related tasks: high temperature, indicating the high randomness of model output. We added this scenario because psychologists argue that there is a positive correlation between randomness and creativity (Campbell 1960), and we take that as a potential mechanism for generative agents to have creative outputs. We document the detailed description of how we parameterized agents for different tasks in the appendix.

3.1 Sparse Autoencoders Setting

We conducted our experiments using Llama-3.3-70B-Instruct and built our experiment platform via Expected Parrot, an open-source app for simulations with generative agents (Horton et al. 2024), where we parameterized the agents for completing tasks and prompted them to boost specific traits. On the interpretability side, we used a set of trained SAEs by Goodfire on the same model trained on LMSYS Chat data (McGrath et al. 2024a, Zheng et al. 2024). These SAEs are trained on layer 50, which is an intermediate position among all the 80 layers, capturing the agent-level behavioral codes instead of just the syntax and perception of agents' conversations. While our SAEs already carry substantial meaning after training, they are not in human-readable forms. Goodfire then uses Claude to interpret the tokens and contexts from the trained SAEs and excludes 3.5% of all the features that are socially harmful, leading to 65,536 features in total (McGrath et al. 2024b).

Figure 1 illustrates the SAE pipeline for both interpretability and steering. Given a prompt, the LLM performs a forward pass up to intervention layer ℓ, yielding a hidden state $\mathbf{h}_\ell \in R^d$. The SAE encoder maps this activation to a sparse latent code:

$$\mathbf{z} = f_{\text{enc}}(\mathbf{h}_\ell)$$

where $\mathbf{z} \in R^k$ is sparse (most entries are zero) and each non-zero dimension corresponds to an

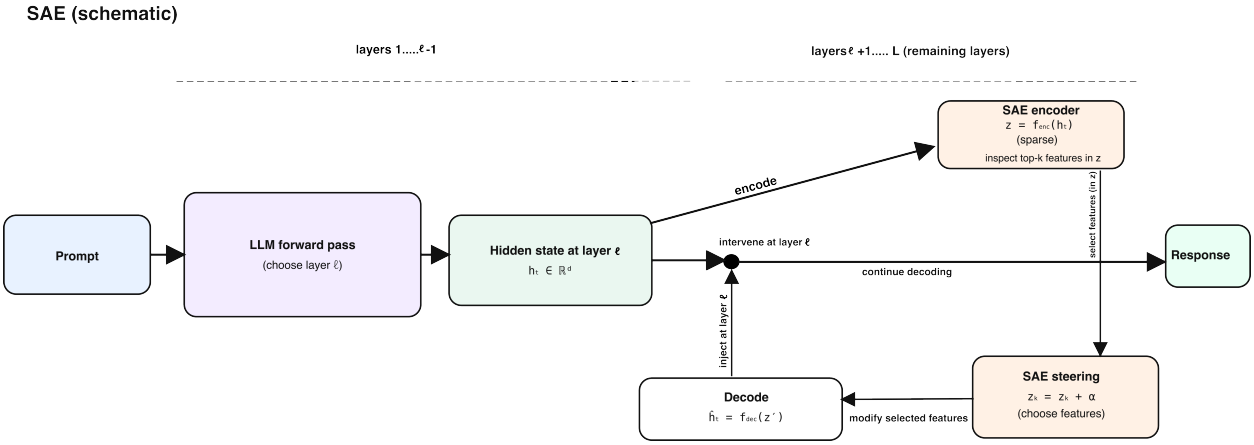


Figure 1: SAE interpretability and steering pipeline. The hidden state $\mathbf{h}_\ell$ at layer ℓ is encoded into a sparse latent $\mathbf{z}$, where top-k features can be inspected for interpretability. For steering, selected features are modified ($\mathbf{z}' = \mathbf{z} + \alpha$) before decoding back to $\mathbf{h}'_\ell$ and continuing generation.

interpretable feature (Bricken et al. 2023). This sparse decomposition addresses the problem of superposition, where neural networks pack multiple features into fewer dimensions, making individual features difficult to isolate (Elhage et al. 2022). For interpretability, we inspect the top-k activated features in $\mathbf{z}$ to understand which behavioral concepts drive the agent's response.

For steering, we modify selected feature activations in the latent space:

$$\mathbf{z}' = \mathbf{z} + \alpha \cdot \mathbf{e}_i$$

where $\mathbf{e}_i$ is a one-hot vector selecting feature i and $\alpha \in [-1, 1]$ controls the steering strength. Negative α dampens the feature while positive α amplifies it. The modified latent is then decoded back to activation space:

$$\mathbf{h}'_\ell = f_{\text{dec}}(\mathbf{z}')$$

This steered activation $\mathbf{h}'_\ell$ is injected at layer ℓ, and decoding continues through the remaining layers to produce the final response. This encode-modify-decode pipeline enables fine-grained control over specific interpretable features while preserving the model's overall coherence.

3.2 Probes Setting

Figure 2 illustrates our probe-based pipeline for both scoring and steering. We train a logistic regression probe on intermediate layer activations (layer 17 of Qwen-2-7B-Instruct) to identify directions associated with agents' traits (Alain and Bengio 2017, Conneau et al. 2018). Layer 17 was selected via a cross-validated sweep over layers and achieved the highest held-out performance (CV score 0.99).

Given a prompt, the LLM performs a forward pass up to intervention layer ℓ, yielding a hidden state $\mathbf{h}_\ell \in R^d$. For scoring, the linear probe computes:

$$\text{score} = \mathbf{w}^\top \mathbf{h}_\ell + b$$

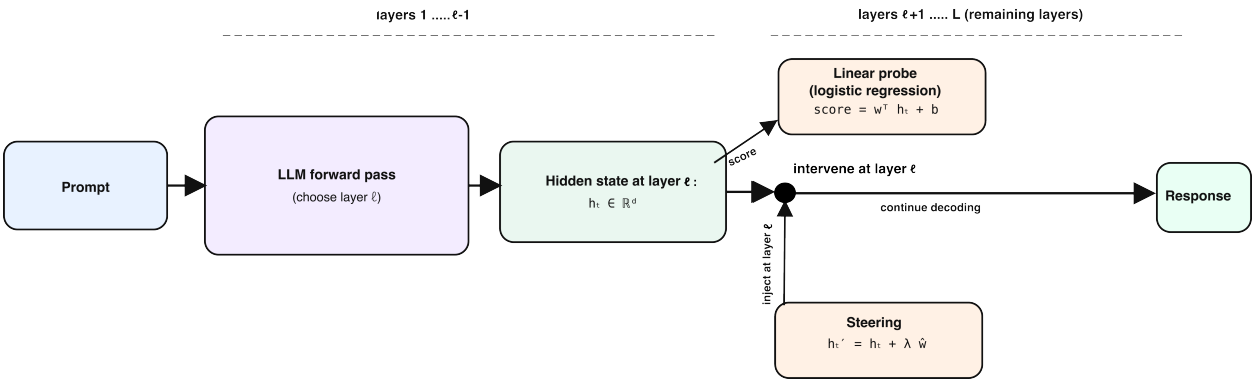


Figure 2: Probe scoring and steering pipeline. The hidden state $\mathbf{h}_\ell$ at layer ℓ feeds into a linear probe that computes a trait score (score $= \mathbf{w}^\top \mathbf{h}_\ell + b$). For steering, the probe direction is added to shift behavior ($\mathbf{h}'_\ell = \mathbf{h}_\ell + \lambda \hat{\mathbf{w}}$) before continuing generation.

where $\mathbf{w} \in R^d$ is the learned probe weight vector and b is the bias term. This score quantifies the model's internal representation along the trait dimension (e.g., risk-seeking vs. risk-averse).

For steering, we add a scaled version of the probe direction to shift the model's behavior:

$$\mathbf{h}'_\ell = \mathbf{h}_\ell + \lambda \cdot \hat{\mathbf{w}}$$

where $\hat{\mathbf{w}} = \mathbf{w}/||\mathbf{w}||_2$ is the unit-normalized probe direction and λ controls the steering strength (Turner et al. 2024). Positive λ pushes the representation toward the probe's positive class (e.g., more risk-seeking), while negative λ pushes toward the negative class. The steered activation $\mathbf{h}'_\ell$ is then injected at layer ℓ, and decoding continues through the remaining layers to produce the final response. We interpret probe scores cautiously, as probe performance can reflect probe capacity rather than the structure of the underlying representations (Hewitt and Liang 2019).

Building probes requires pre-defined features of agents to be measured. Hence, we generated synthetic data controlled on the different degrees of the selected features as training data. For the preference tasks, we generated training data by getting agents' responses to lottery and ultimatum game scenarios, extracting activations at layer 17, and assigning binary labels based on median-split criteria (high vs. low risk-taking for lottery, high vs. low altruism for ultimatum). The probe achieved 82% classification accuracy on held-out trials, suggesting that preference-related information is linearly accessible in the model's representations. For creativity tasks, we employed a separate LLM judge (Llama-3-8B-Instruct) to score creative outputs on a 1-10 scale, then trained a probe to distinguish high-creativity (above median) from low-creativity responses. This independent evaluation architecture ensures that the steering model is not "grading its own homework," addressing potential circularity concerns in capability assessment.

Task Type	Intervention / Setting			
	Baseline	Prompting	Steering	Temperature
Lottery Game	-	Risky persona	2 risk-related SAE features boosted	0.5
Ultimatum Game	-	Altruistic persona	1 altruism SAE feature boosted	0.5
Divergent Creativity	-			
Product Innovation	-			

Table 2: Interventions used for preference vs. capability tasks.

4 Results

We evaluate how different tools perform on preference and capability tasks in terms of their interpretability and controllability over agents' behaviors. First, we showcase the top-ranked SAE features decomposed by the agents' responses in different settings to demonstrate the interpretability of their behaviors. Table 2 details the interventions we have introduced to the agents for preference and capability tasks. In baseline scenarios, agents respond according to the task descriptions without interventions. In prompting scenarios, agents are prompted with different personas, specified by the known traits about the specific tasks. In steering scenarios, we manually adjust the activation strength of some specified related SAE features to alter the agents' behaviors. In high-temperature scenarios, which are designed specifically for capability tasks, agents are configured to produce more random responses to assess whether increased randomness leads to higher creativity. With the results from all the scenarios, inspecting the SAE features of generative agents allows us to peek into the behavioral codes of generative agents.

Second, we showcase how prompting, SAE steering, and probe steering differ in their performances on the controllability of agents' behaviors. We provide different degrees of control in each approach as separate scenarios and show the agility in the agents' responses. For each scenario, we repeat the same steering interventions on 40 agents. In preference tasks, we record the percentage of agents changing their behaviors as reward values increase in the lottery and the ultimatum games; and in capability tasks, we record the average creativity scores with standard deviation in divergent creativity and product innovation tasks.

4.1 Interpretability

4.1.1 Preference Tasks

Figure 3 details the top 10 features activated for generative agents in preference tasks. We observe that most of the top 3 features show no variation in their minimum and maximum ranks. For the lottery game, the top 3 features are consistent across the baseline and prompting scenarios. "Probability-based decision making scenarios with explicit numerical comparisons" and "The assistant should select between provided options" occupy top 1 and 2 features for all agents in both scenarios; and "Technical setup and configuration states in experimental procedures" takes the 3rd place for most agents with few in the 4th place. High rankings of these features reinstate the underlying theory for the lottery game, for which agents use probability-based decision making to support rational choices. This result replicates the existing social theories and behavioral principles.

However, the rest of the features are less common. For example, "The assistant is explaining why it cannot fulfill an unethical request" is a feature that provides less intuition in making decisions in the lottery games. Inspecting features like this can potentially inspire social scientists to add another layer of analysis beyond rational choices and probabilistic decision-making, which is to consider ethics in the lottery games.

For the ultimatum game, the top 3 features show a similar lack of variation in ranks across baseline and prompting scenarios and disclose the underlying mechanisms of agents' behaviors. "Economic tradeoffs and payoff structures in social contexts", "Game theory concepts involving cooperation versus competition", and "Formal offer and counteroffer language in negotiations" confirm that agents grasp the essence of the ultimatum game in leveraging altruism and selfish decision-making principles in economic games. Other lower-ranked features, such as "Descriptions of experimental procedures and participant tasks in research studies," do not have a direct link with the existing theories about the ultimatum game but rather with the methodology we implement on the agents. This shows that SAE features not only capture the mechanisms underlying specific tasks, but also the agents behavioral codes in general.

Identifying SAE features also helps investigate the effects of controllability. In the steering scenario for both games, the top-ranked feature becomes the feature we manually steered, despite not being in the top-ranked feature list for the baseline scenario. The rest of the features are still similar to the ones in the baseline and prompting scenarios. This confirms the effects of feature steering as it mechanistically changes the underlying behavioral codes of generative agents. We can also observe that the top-ranked features for the prompting scenario are more similar to the features for the baseline scenario rather than steering scenario.

4.1.2 Capability Tasks

Figure 4 compares the top five activated features within generative agents across two task types, divergent creativity and product innovation, under four conditions: baseline, high temperature, prompting, and steering. In the baseline setting, the model activates reasonable but relatively generic features, such as brainstorming lists or describing technical components. Increasing temperature introduces noise, with features tied to messy formatting, multilingual fragments, and irregular text, indicating exploration without control. Prompting makes behavior more focused, aligning activations more closely with relevant concepts like creative alternatives and technical relationships, though still indirectly. By contrast, steering produces the strongest and most targeted activations, concentrating on concepts such as professional innovation, unconventional thinking, and structured feature descriptions. Together, these patterns show that feature activations decomposed by SAEs are useful for understanding the mechanisms supporting agentic behaviors.

Two practical implications of this behavioral interpretability include theoretical validation and generation. For example, if one hypothesizes the creativity related to innovation can be extended from considering the physical restraints of the current products, the top 1 or 2 SAE features activated for agents in the product innovation tasks confirm this hypothesis. However, while the SAE features offer some extent of validation, we need to carefully evaluate their activation strength and underlying logic. One counterexample of this is the second SAE feature activated for divergent creativity tasks, "The Russian word for brick", which does not offer intuitive mechanisms of creativity but rather can be seen more as a linguistic hint within language models. Theoretical generation is also possible: we can use the top-ranked activated features to brainstorm mechanisms that may underlie

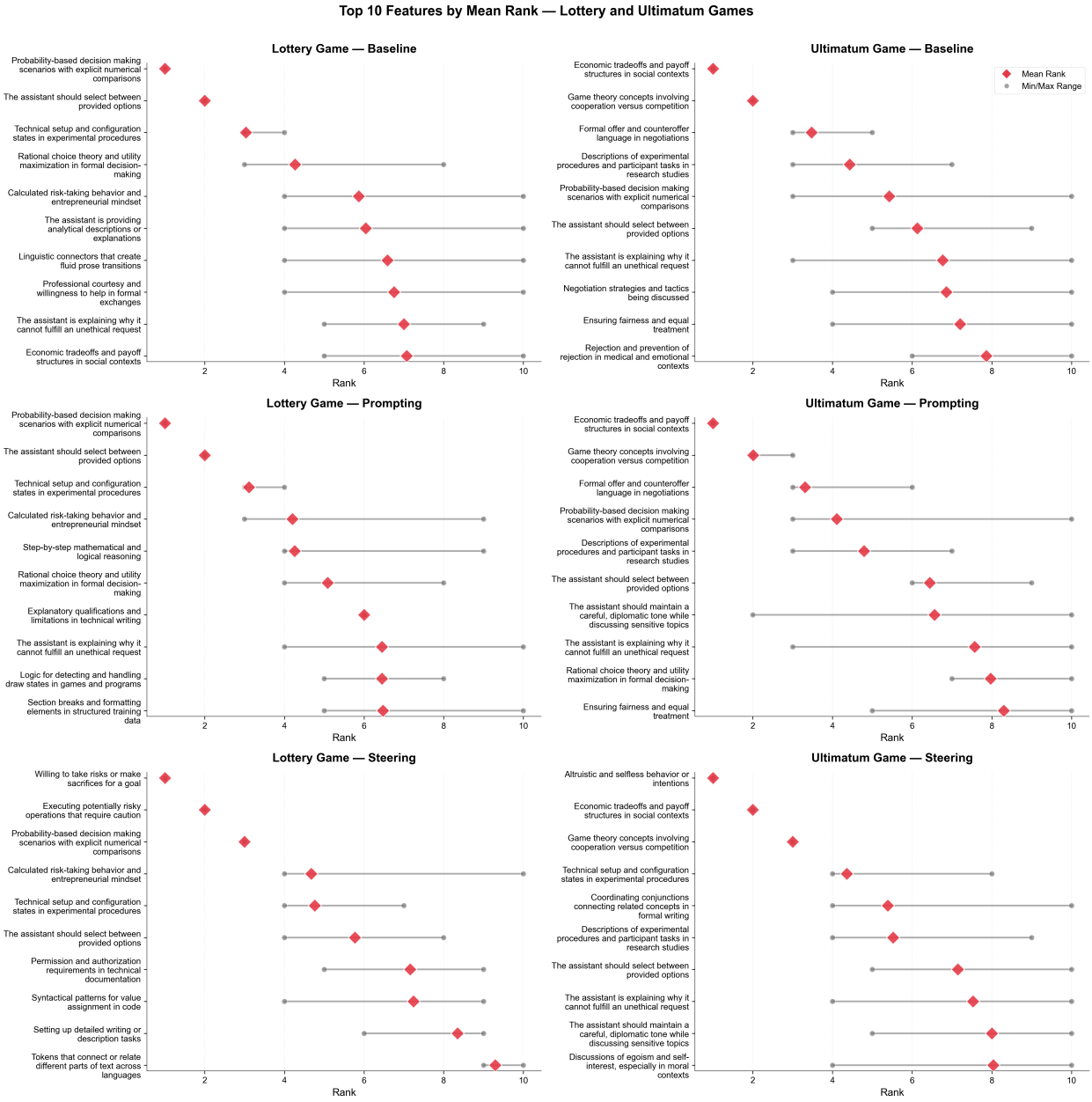


Figure 3: Feature Activations of Generative Agents on Preference Tasks

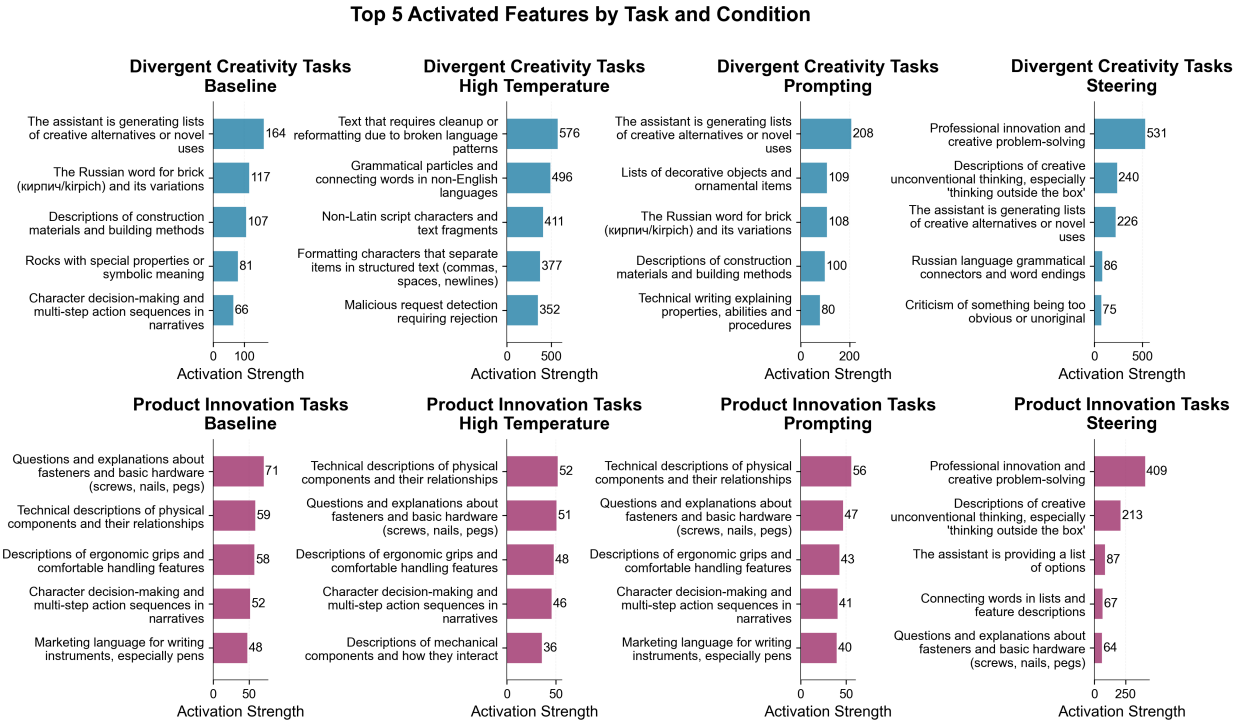


Figure 4: Feature Activations of Generative Agents on Capability Tasks

agentic behaviors, even if those mechanisms are not yet theoretically grounded in established social-scientific behavioral patterns. For example, "marketing language for writing instruments" is not an usual behavioral code for a stapler innovation task. This suggests that creativity in product innovation may stem from connecting the focal product to others that appear in the same context. With this example, we demonstrate that investigating SAE features is beneficial for inductive theorizing.

4.2 Controllability

After characterizing the interpretability of reasoning that underlies agents' behaviors, we next examine whether, and to what extent, these behaviors can be steered. To this end, we leverage all three approaches: prompting, probe-based methods, and sparse autoencoders.

4.2.1 Preference Tasks

We first use the lottery game to demonstrate risk-taking behaviors of humans. According to our setting, the equilibrium between risky and safe options is for the risky reward value to be equal to 100, when the expected value of risky option is the same as the safe option. In the baseline scenario demonstrated in the left panel of Figure 5, generative agents exhibit human-like risk aversion, shifting their preference only once the risky reward exceeds 100 tokens, around 125. In the prompting scenario where we explicitly prompt the agents to be risk-taking in their traits, they start pivoting to risky option when the reward is only 20 tokens. We also observe changes in the steering scenario, where we identify the SAE features responsible for risky behaviors and manually boost

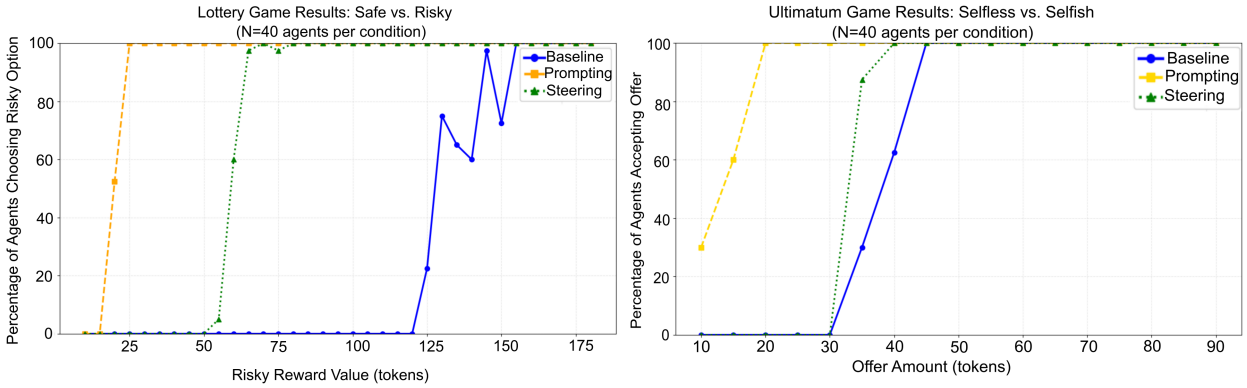


Figure 5: Performance of Generative Agents on Preference Tasks

them for outputs. However, agents only start to pivot to risky option when risky rewards become more than 50. This indicates that prompted agents are more risk-taking than steering agents.

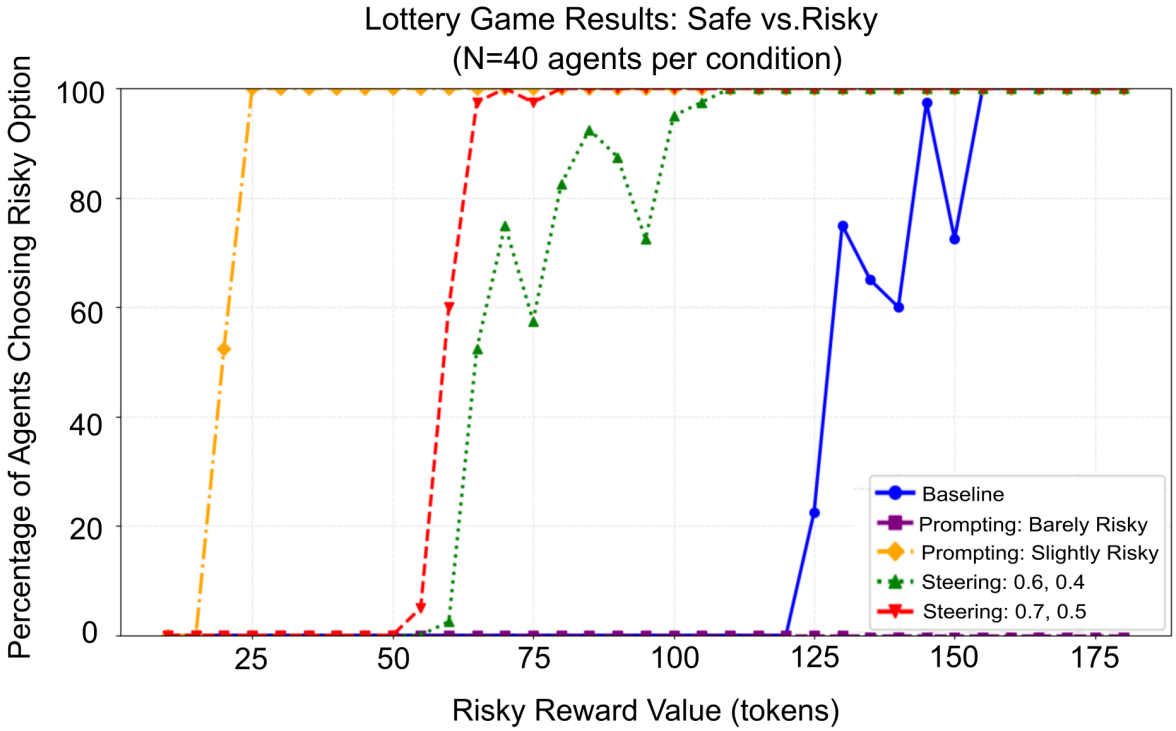


Figure 6: Lottery game results across five experimental conditions.

We extrapolate further about what is behind this phenomenon and we find that steering controls the agents' behaviors in a higher granularity than prompting. As shown in Figure 6, slight changes in prompting language could lead to big yet unpredictable results in agents' behaviors. In contrast, steering with different activation strengths lead to smoother changes in when the agents start to take risky options. The baseline shows risk aversion with agents switching to risky option around 125 tokens. Prompting variants demonstrate binary behavior: "barely risky" shows no effect, while "slightly risky" produces extreme risk-seeking even at very low rewards (20-25 tokens). In contrast,

SAE steering conditions show dose-dependent, controllable effects: moderate steering (activations 0.6/0.4, green dotted) produces gradual transition starting around 75 tokens, while stronger steering (activations 0.7/0.5, red dashed) shows sharper but still controlled transition around 65-70 tokens. This demonstrates that SAE steering provides fine-grained control over agent behavior, whereas prompting produces unpredictable all-or-nothing effects.

Probe-based steering also demonstrates successful behavioral control in the lottery task, as demonstrated in Figure 7. We achieve precise control over risk preferences by calibrating λ values to target specific switching points, defined as the risky reward value at which the probability of choosing the risky option equals 0.5. The baseline agent exhibits risk aversion with a switching point around 125 tokens. Through systematic search over $\lambda \in [-6, 6]$, we successfully shift this switching point across a wide range from 40 to 120 tokens, with achieved values closely matching targets (mean absolute error of 3.2 tokens). Higher positive λ values increase risk-seeking behavior (lower switching points), while negative values enhance risk aversion.

We demonstrate that ultimatum game has similar results to lottery game in how prompting and steering could effectively change the behaviors of generative agents. More specifically, our setting of the ultimatum game makes an even share of the tokens at 50. As shown in the right panel of Figure 5, the baseline scenario demonstrates that the generative agents behave slightly more altruistically than perfectly rational humans, where they start to take the offer when it exceeds 30. We prompt the agents to have altruism or selfless behaviors in the prompting scenario and the effects are outstanding, as they start to take the offer even when the offer is only 10 tokens. Panel (b) of Figure 7 for using probes to alter agents' behaviors in ultimatum games also demonstrate similar patterns. It shows that the acceptance threshold moves from about 30 tokens at baseline up to 60 tokens under stronger steering. Curves are approximately monotonic with diminishing returns at extreme λ values, indicating saturation of the intervention.

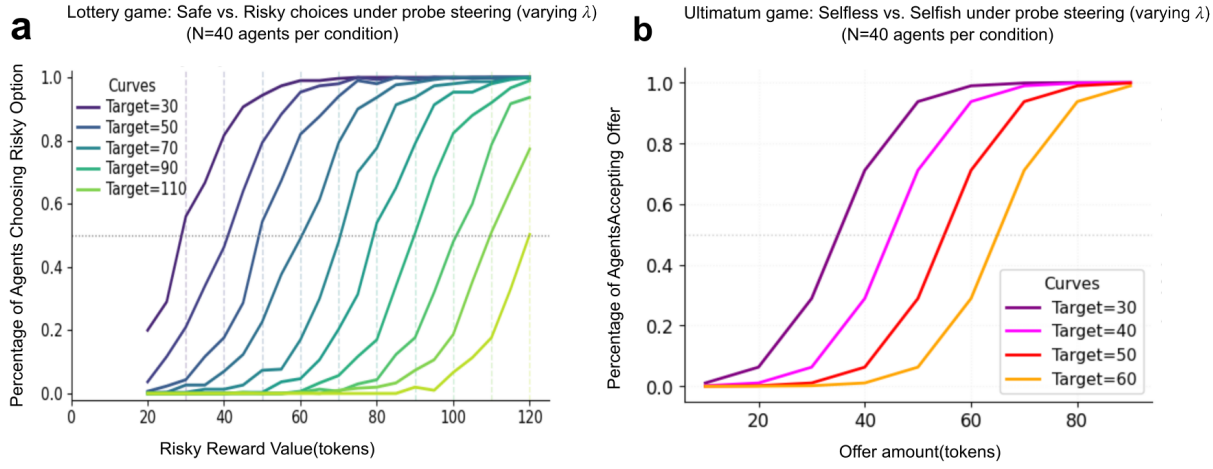


Figure 7: (a) & (b) psychometric curves across calibrated λ values for lottery and ultimatum games

4.2.2 Capability Tasks

We implement the divergent creativity and product innovation tasks on generative agents and evaluate their outputs with GPT-5 (OpenAI 2025) and Llama-3-8B-Instruct (Llama Team 2024),

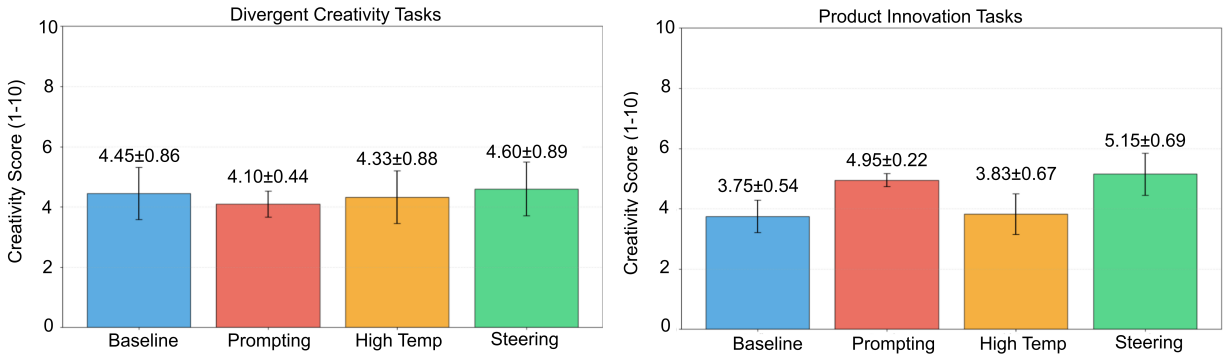


Figure 8: Performance of Generative Agents on Capability Tasks

based on their fluency, flexibility, originality, and elaboration (Zhao et al. 2025).[1] Figure 8 shows that steering SAE features produces the largest increase in creativity from the baseline scenario across both tasks. In contrast, prompting the creative traits into the personas of generative agents does not always guarantee positive results in boosting their outputs' results. Similarly, agents with high randomness to produce their outputs also have unpredictable changes in the degree of creativity. These observations suggest that if we want to use creativity as a variable for theorizing capabilities deductively, SAE features are the only relatively legitimate approach here. Stimulating the agents mechanistically through their behavioral codes, steering SAE features effectively causes output changes.

Figure 9 presents controllability results for product innovation with different objects in Torrance-style "uses" prompts (brick, stapler, paperclip, bowl) via our probe-based methods. Each panel shows the achieved creativity score as a function of the target score. Error bars represent standard error across 48 trials. The diagonal line indicates perfect controllability. Unlike the preference tasks, capability control through probes shows more limited dynamic range. Despite systematic search over λ values, achieved creativity scores cluster between 6.7 and 7.8 on the 10-point scale, regardless of target values. This ceiling effect occurs consistently across all four objects, suggesting fundamental limitations in using single linear directions to control complex, multi-faceted capabilities like creativity.

The restricted range is particularly evident when targeting low creativity scores (3 or 5). The probe-steered model cannot produce genuinely uncreative outputs, with minimum achieved scores around 6.7. This asymmetry, specifically the inability to reduce capability below baseline while showing modest increases, aligns with the probe's training on median-split labels rather than continuous regression targets. The high baseline scores (approximately 7.3 without steering) leave limited headroom for improvement through positive steering.

[1]Definitions of the 4 criteria: 1. Fluency: The ability to produce a significant number of relevant ideas in response to a given question. 2. Flexibility: The variety of categories from which one can generate ideas. 3. Originality: The uniqueness of the ideas generated. 4. Elaboration: The ability to expand upon, refine, and embellish an idea.

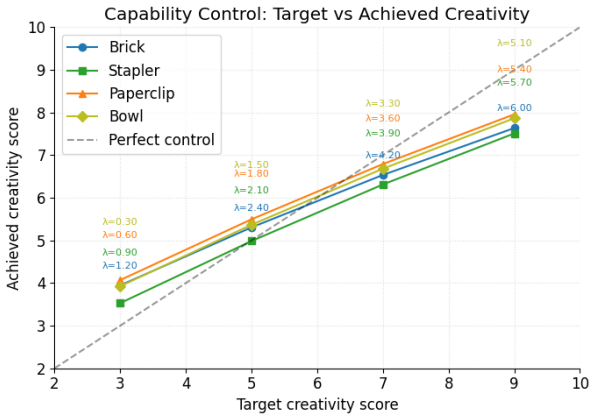


Figure 9: Achieved creativity scores as a function of target scores across four object prompts, showing limited controllability range.

5 Discussion

Our results have demonstrated the practicality of using prompt-based, SAE and probe-based tools to achieve both interpretability and controllability in social scientific experiments. To translate these results into actionable methodological choices, we first compare these approaches on their strengths and trade-offs, and then turn to the boundary conditions and limitations that shape how they should be applied in practice.

First, prompt-based methods have drawbacks of not fully capturing the underlying behavioral mechanisms and is ineffective at steering behaviors compared to the other two. This is because generative agents are auto-regressive in nature, which make them justify what they did than showcase their actual rationale behind behaviors. This indicates that the behaviors we observe from prompts reflect narrative explanations, not the underlying processes that produced those actions, limiting prompting's inferential validity. In addition, human instructions may not align with the way agents interpret and represent those instructions internally. This misalignment is two-fold: one is on the misunderstanding of natural language and another is on the structural deficiency of agents to encode instructions from humans.

Second, SAE-based methods achieve better controllability than probe-based methods in multi-feature represented behaviors. For example in our creativity tasks, SAE-based methods can capture the fluency, flexibility, originality, and elaboration aspects and coin them together to compose creativity. However, this process is hard for probe-based methods to capture as it resists compression into a single linear direction.

Third, The probe approach offers computational advantages: training requires only hundreds of examples rather than the massive datasets needed for SAEs, and inference adds minimal overhead compared to SAE encoding/decoding. However, this efficiency comes at the cost of expressiveness. Probes provide reliable behavioral control when the target trait aligns with a dominant linear direction in activation space but struggle with complex capabilities requiring coordinated changes across multiple semantic dimensions.

These findings suggest some practical guidance for method selection: computational social scientists should choose the methods depending on the goals of their research. If the goal of research is to inductively investigate hypotheses behind a complex real-world phenomenon, then

SAE-based mechanistic interpretability tools will uncover the internal links behind behavioral codes. If deductive experiments on pre-defined measurable traits are the targets, probe-based methods provide more fine-grained controllability efficiently.

While this paper provides one of the first attempts to use interpretability tools in social experiment simulations, the comparisons above also clarify several limitations that constrain what these methods can currently guarantee.

First, misuse of SAE-based methods can introduce unintended and potentially detrimental side effects. The extent to which we should steer an SAE feature to create a desired LLM persona for experimentation is more art than standardized practice. Steering too much will lead to hallucination, which produces nonsensical output. Furthermore, as SAE features are not perfectly disentangled, boosting one feature too much may activate other off-target representations, causing behavioral shift that are hard to analyze. These risks highlight the need for careful monitoring and calibration while using steering in simulating experiments.

Second, we limit our focus to individuals in this paper. While individuals are the most fundamental component of society, many experiments do not happen on the individual level. This leads to a natural thread of future work in multi-agent settings, where generative agents will interact with others in structured social environments, allowing emerging group dynamics to be studied. Such extensions would explore organizational structure, social influence, and network effects, which are only possible to be accessed in multi-agent settings. This is crucial for advancing generative agent simulations towards a richer and more ecologically valid theorizing process.

Third, our experiments focus on a small set of traits developed from preference and capability tasks. They map directly to behavioral constructs. However, many social scientific mechanisms are more abstract, context-dependent, or embedded within complex structures. Future research in using SAE features to capture higher-order mechanisms need to examine the mechanistic support carefully in order to produce robust theorizing results.

Future work might also explore hybrid approaches, such as multi-probe steering or low-rank adaptations, that balance computational efficiency with representational capacity. Together, these limitations constitute a comprehensive set of future research trajectories, indicating that using mechanistically interpretable generative agents in computational social science is promising.

6 Conclusion

Generative agents powered by LLMs give computational social scientists a scalable, low-cost way to simulate behavior, but their usefulness is limited by the black-box nature of LLM reasoning. We show that interpretability techniques, especially SAEs and probes, can open this black box by identifying and steering behaviorally meaningful features, enabling more controlled agent decision-making in simulations.

We frame preferences and capabilities as two core domains of human behavior and show that SAE-based and probe-based steering produces results that are more stable, interpretable, and controllable than prompt-based methods. This allows SAEs and probes to support both inductive and deductive theorizing, enriching hypothesis development while improving causal validity in simulated experiments.

Our results establish that investigating internal mechanisms is essential for rigorous theory building with generative agents. We hope to use this paper as a primer to leverage LLM personas to

accelerate discovery in the social sciences via computational experimentation.

References

[1] Abdulhai, Marwa, Gregory Serapio-García, Clément Crépy, Daria Valter, John Canny, and Natasha Jaques. 2024. "Moral Foundations of Large Language Models." In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[2] Alain, Guillaume, and Yoshua Bengio. 2017. "Understanding Intermediate Layers Using Linear Classifier Probes." *ICLR Workshop*.

[3] Anthis, Jacy Reese, Ryan C. Liu, Sean M. Richardson, Austin C. Kozlowski, Bernard Koch, James A. Evans, Erik Brynjolfsson, and Michael S. Bernstein. 2025. "LLM Social Simulations Are a Promising Research Method." *arXiv preprint* arXiv:2504.02234.

[4] Belinkov, Yonatan. 2022. "Probing Classifiers: Promises, Shortcomings, and Advances." *Computational Linguistics* 48(1):207–219.

[5] Bender, Emily M., Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*:610–623.

[6] Bricken, Trenton, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. 2023. "Towards Monosemanticity: Decomposing Language Models with Dictionary Learning." Anthropic Technical Report.

[7] Camerer, Colin, and Richard H. Thaler. 1995. "Anomalies: Ultimatums, Dictators and Manners." *Journal of Economic Perspectives* 9(2):209–219.

[8] Campbell, Donald T. 1960. "Blind Variation and Selective Retention in Creative Thought." *Psychological Review* 67(6):380–400.

[9] Chang, Serina, Alicja Chaszczewicz, Emma Wang, Maya Josifovska, Emma Pierson, and Jure Leskovec. 2025. "LLMs Generate Structurally Realistic Social Networks but Overestimate Political Homophily." *arXiv preprint* arXiv:2408.16629.

[10] Chen, Hui, Antoine Didisheim, Luciano Somoza, Hanqing Tian 2025. "A Financial Brain Scan of the LLM." *arXiv preprint* arXiv:2508.21285

[11] Chen, Yiting, Tracy Xiao Liu, You Shan, and Songfa Zhong. 2023. "The Emergence of Economic Rationality of GPT." *Proceedings of the National Academy of Sciences* 120:e2316205120.

[12] Conneau, Alexis, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. "What You Can Cram into a Single $&!#* Vector: Probing Sentence Embeddings for Linguistic Properties." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*:2126–2136.

[13] Davis, Jason P., and Christopher B. Bingham. 2007. "Developing Theory Through Simulation Methods." *Academy of Management Review* 32(2):480–499.

[14] Edwards, Ward. 1954. “The Theory of Decision Making.” *Psychological Bulletin* 51(4):380–417.

[15] Elhage, Nelson, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. "A Mathematical Framework for Transformer Circuits." Anthropic Technical Report.

[16] Elhage, Nelson, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. "Toy Models of Superposition." Anthropic Technical Report.

[17] Epstein, Joshua M. 1999. “Agent-Based Computational Models and Generative Social Science.” *Complexity* 4(5):41–60.

[18] Gao, Chen, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. 2024. “Large Language Models Empowered Agent-Based Modeling and Simulation: A Survey and Perspectives.” *Humanities and Social Sciences Communications* 11:1259.

[19] Grimmer, Justin, Margaret E. Roberts, and Brandon M. Stewart. 2021. "Machine Leaning for Social Science: An Agnostic Approach." *Annual Review of Political Science* 24:395-419

[20] Hedström, Peter. 2005. *Dissecting the Social: On the Principles of Analytical Sociology*. Cambridge: Cambridge University Press.

[21] Hewitt, John, and Christopher D. Manning. 2019. "A Structural Probe for Finding Syntax in Word Representations." *Proceedings of NAACL-HLT*.

[22] Hewitt, John, and Percy Liang. 2019. "Designing and Interpreting Probes with Control Tasks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*:2733–2743.

[23] Horton, John J. 2023. “Large Language Models as Simulated Economic Agents: What Can We Learn from Homo Silicus?” *NBER Working Paper* 31122.

[24] Horton, John J., Apostolos Filippas, and Rose Horton. 2024. “EDSL: Expected Parrot Domain Specific Language for AI Powered Social Science.” Expected Parrot Whitepaper.

[25] Jackson, Matthew O., Qiaozhu Mei, Stephanie Wang, Yutong Xie, Walter Yuan, Seth G. Benzell, Erik Brynjolfsson, Colin F. Camerer, James A. Evans, Brian, Jabarian, Jon Kleinberg, Juanjuan Meng, Sendhil Mullainathan, Asuman E. Ozdaglar, Thomas Pfeiffer, Moshe Tennenholtz, Robb Willer, Diyi Yang, and Teng Ye. 2025. “AI Behavioral Science.” *arXiv preprint* arXiv:2509.13323.

[26] Kazinnik, Sophia. 2023. "Bank Run, Interrupted: Modeling Deposit Withdrawals with Generative AI." Available at *SSRN*: http://dx.doi.org/10.2139/ssrn.4656722

[27] King, Gary. 2011. “Ensuring the Data Rich Future of the Social Sciences”. *Science*, 331:719-21.

[28] Lazer, David, Alex Pentland, Lada Adamic, Sinan Aral, Albert-László Barabási, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, Tony Jebara, Gary King, Michael Macy, Deb Roy, and Marshall Van Alstyne. 2009. “Computational Social Science.” *Science* 323(5915):721–723.

[29] Llama Team (Grattafiori, Anthony, et al. (Meta AI)). 2024. “The Llama 3 Herd of Models.” *arXiv preprint* arXiv:2407.21783.

[30] Ludwig, Jens, and Sendhil Mullainathan. 2024. "Machine Learning as a Tool for Hypothesis Generation." *The Quarterly Journal of Economics* 139(2): 751-827

[31] Macy, Michael W., and Robb Willer. 2002. “From Factors to Actors: Computational Sociology and Agent-Based Modeling.” *Annual Review of Sociology* 28:143–166.

[32] McGrath, Thomas, Daniel Balsam, Myra Deng, and Eric Ho. 2024. “Understanding and Steering Llama 3 with Sparse Autoencoders.” Goodfire Research Report.

[33] McGrath, Thomas, Daniel Balsam, Liv Gorton, Murat Cubuktepe, Myra Deng, Nam Nguyen, Akshaj Jain, Thariq Shihipar, and Eric Ho. 2024. “Mapping the Latent Space of Llama 3.3 70B.” Goodfire Research Report.

[34] Nanda, Neel. 2023. “Attribution Patching: Activation Patching at Industrial Scale.” *Transformer Circuits Research Report*.

[35] OpenAI. 2025. *GPT-5 System Card*. San Francisco, CA: OpenAI.

[36] Park, Joon Sung, Carolyn Q. Zou, Aaron Shaw, Benjamin Mako Hill, Carrie J. Cai, Meredith Ringel Morris, Robb Willer, Percy Liang, and Michael S. Bernstein. 2024. “Generative Agent Simulations of 1,000 People.” *arXiv preprint* arXiv:2411.10109.

[37] Rudin, Cynthia. 2019. “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead.” *Nature Machine Intelligence* 1(5):206–215.

[38] Salganik, Matthew J. 2017. *Bit by Bit: Social Research in the Digital Age*. Princeton, NJ: Princeton University Press.

[39] Torrance, E. Paul. 1966. *Torrance Tests of Creative Thinking: Norms-Technical Manual*. Princeton, NJ: Personnel Press.

[40] Turner, Alexander Matt, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2024. "Activation Addition: Steering Language Models Without Optimization." *arXiv preprint* arXiv:2308.10248.

[41] Toubia, Olivier, George (Zhida) Gui, Tianyi Peng, Daniel J. Merlau, Ang Li, and Haozhe Chen. 2025. “Twin-2K-500: A Data Set for Building Digital Twins of over 2,000 People Based on Their Answers to over 500 Questions.” *Marketing Science*. Forthcoming.

[42] Tranchero, Matteo, Cecil-Francis Brenninkmeijer, Arul Murugan, and Abhishek Nagaraj. 2024. “Theorizing with Large Language Models” *NBER Working Paper* 33033.

[43] Turpin, Miles, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. “Language Models Don’t Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting.” *arXiv preprint* arXiv:2305.04388.

[44] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” *Advances in Neural Information Processing Systems* 30:5998–6008.

[45] Xie, Yutong, Qiaozhu Mei, Walter Yuan, and Matthew O. Jackson. 2025. "Using Large Language Models to Categorize Strategic Situations and Decipher Motivations Behind Human Behaviors." *Proceedings of the National Academy of Sciences* 122:e2512075122.

[46] Zhao, Yunpu, Rui Zhang, Wenyi Li, and Ling Li. 2025. "Assessing and Understanding Creativity in Large Language Models." *Machine Intelligence Research*. https://doi.org/10.1007/s11633-025-1546-4

[47] Zheng, Lianmin, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2024. "LMSYS-Chat-1M: A Large-Scale Real-World LLM Conversation Dataset." *arXiv preprint* arXiv:2309.11998.

A Technical Appendix for Prompting and Sparse Autoencoder Setting

A.1 Stack Overview

EDSL orchestration. We use Expected Parrot's EDSL to define each game as a typed interaction with strict I/O schemas. In the safe-vs-risky game we use:

- `QuestionMultipleChoice` with options `["Safe Option", "Risky Option"]`
- An attached `*_comment` field to capture the model's rationale
- Built-in validators so malformed outputs are reprompted consistently

In code (from `SafeRiskyChoice.setup()`):

```
choice_text = (
    "SAFE VS. RISKY CHOICE GAME\n\n"
    "You must choose between two options:\n"
    f"1. Safe Option: Guaranteed 50 tokens.\n"
    f"2. Risky Option: 50% chance of {self.risky_reward} tokens,
       50% chance of 0 tokens.\n\n"
    "Which option do you choose?"
)
self.questions.append(
    QuestionMultipleChoice(
        question_name="safe_risky_choice",
        question_text=choice_text,
        question_options=["Safe Option", "Risky Option"],
    )
)
```

Goodfire interpretability. We use Goodfire's OpenAI-compatible API to:

1. call `meta-llama/Llama-3.3-70B-Instruct` as the base model
2. read SAE feature activations on demand

3. apply feature-level "nudge" steering during decoding

We instantiate both the EDSL model and a Goodfire Variant:

```
import goodfire
client = goodfire.Client(os.getenv("GOODFIRE_API_KEY"))
base_variant = goodfire.Variant("meta-llama/Llama-3.3-70B-Instruct")

# EDSL model routed through Goodfire
model = Model("meta-llama/Llama-3.3-70B-Instruct",
              service_name="goodfire")
```

EDSL ↔ Goodfire bridge. The bridge maps EDSL's `.parameters["controller"]` to Goodfire's controller schema. When empty, we run the base model. When populated, Goodfire applies feature-level steering. We reset the Variant when turning steering off to avoid bleed-through.

A.2 Agents, Personas, and Cohorts

We run three cohorts that differ only in how behavior is induced:

Baseline agents: neutral personas

```
dummy_agents = AgentList([
    Agent(name=f"Agent_{i}",
          traits={"subject_id": f"A{i}"},
          instruction="")
    for i, age in enumerate(range(20, 60, 1), start=1)
])
```

Prompting agents: natural-language trait aligned to "slight risk preference"

```
slightly_risk_agents = AgentList([
    Agent(name=f"SlightlyRisk_Agent_{i}",
          traits={"subject_id": f"SR{i}",
                  "personality": "slightly likes to take risks"},
          instruction="")
    for i, age in enumerate(range(20, 60, 1), start=1)
])
```

Steering agents: same as baseline (neutral persona), but we attach a Goodfire controller at decode time (details in §A.3.3).

All cohorts share identical decoding and validation settings. Caching is disabled for runs that collect activations (`cache=False`) to ensure fresh generations.

A.3 Scenarios and How We Implement Them

A.3.1 Baseline

Meaning. Measure unmodified model behavior under our EDSL scaffold.

Implementation.

- Persona: neutral (baseline agents)
- Controller: unset

```
self.model.parameters["controller"] = {}
self.variant = goodfire.Variant(
    "meta-llama/Llama-3.3-70B-Instruct")
```

- Run: `Survey(self.questions)`
  `.by(self.agents).by(self.model)`
  `.run(cache=False)`

What we log.

- CSV per reward level: `safe_risky_baseline_{reward}.csv`
- Columns include `answer.safe_risky_choice` and `comment.safe_risky_choice_comment`
- Optionally we also compute activations against the base variant for analysis (see §A.6)

A.3.2 Prompting

Meaning. Induce similar behavior using only natural-language persona traits.

Implementation.

- Persona: "slightly likes to take risks" (see `slightly_risk_agents`)
- Controller: unset (no steering)
- Everything else identical to baseline

What we log.

- CSV per reward: `safe_risky_slightly_prompting_{reward}.csv`
- Same columns and validators

A.3.3 Steering

Meaning. Induce behavior by nudging human-readable SAE features during decoding, while keeping the persona neutral.

Feature selection. We shortlist risk-related features from Goodfire's 65k SAE library based on label semantics and example firings. In the notebook we use features such as:

Lottery Game:

- "Willing to take risks or make sacrifices for a goal" (`index_in_sae=184`)
- "Executing potentially risky operations that require caution" (`index_in_sae=4237`)

Controller config. We define two presets:

- Standard (`get_intervention`): stronger nudges (e.g., value 0.5 to 0.7)
- Lite (`get_lite_intervention`): smaller magnitudes (e.g., value 0.4 to 0.6)

Schema shape used by the bridge:

```
controller = {
  "interventions": [
    {
      "mode": "nudge",
      "features": {
        "features": [
          {"uuid": "...",
           "label": "...",
           "index_in_sae": 184,
           "max_activation_strength": 1},
          {"uuid": "...",
           "label": "...",
           "index_in_sae": 4237,
           "max_activation_strength": 1},
        ]
      },
      "value": 0.5  # signed magnitude
    },
    # stack multiple nudges if needed
  ],
  "scopes": [],
  "name": "controller__...",
  "nonzero_strength_threshold": None
}
self.model.parameters["controller"] = \
    controller
```

What we log.

- CSV per reward: `safe_risky_steering_{reward}.csv`
- We also store which feature UUIDs and weights were active

A.4 Ultimatum Game Implementation

The ultimatum game follows the same three-scenario structure as the lottery game (Baseline, Prompting, Steering) to measure altruism and fairness preferences.

A.4.1 Task Setup

In code (from `UltimatumGame.setup()`):

```
responder_text = (
    "ULTIMATUM GAME - RESPONDER\n\n"
    "You are participating in an economic decision-making "
    "experiment called the Ultimatum Game.\n\n"
    "Situation:\n"
    "1. Another participant (the proposer) was given 100 tokens "
    "(which have real monetary value).\n"
    f"2. The proposer has decided to offer you {self.offer_amount} "
    "tokens out of their 100 tokens.\n"
    f"3. If you accept this offer, you will receive "
    f"{self.offer_amount} tokens, and the proposer will keep "
    f"{100 - self.offer_amount} tokens.\n"
    "4. If you reject this offer, both of you will receive "
    "0 tokens.\n\n"
    "Do you accept or reject this offer?"
)
self.questions.append(
    QuestionMultipleChoice(
        question_name="ultimatum_response",
        question_text=responder_text,
        question_options=["Accept", "Reject"],
    )
)
```

A.4.2 Agent Cohorts

Baseline agents: neutral personas (same as lottery game)

```
dummy_agents = AgentList([
    Agent(name=f"Agent_{i}",
          traits={"subject_id": f"A{i}"},
          instruction="")
    for i, age in enumerate(range(20, 60, 1), start=1)
])
```

Prompting agents: altruistic/fairness-focused persona

```
fairness_agents = AgentList([
    Agent(name=f"Fairness_Agent_{i}",
          traits={"subject_id": f"F{i}",
                  "personality": "Altruistic and selfless "
                                 "behavior or intentions"},
          instruction="")
    for i, age in enumerate(range(20, 60, 1), start=1)
])
```

A.4.3 Steering Configuration

Feature selection. We use an altruism-related feature from Goodfire's SAE library:

- "Altruistic and selfless behavior or intentions" (`index_in_sae=31935`)

Controller config. Schema shape for ultimatum game steering:

```
controller = {
  "interventions": [
    {
      "mode": "nudge",
      "features": {
        "features": [
          {"uuid": "a239616d4dcf470497b32ad3500f0145",
           "label": "Altruistic and selfless behavior "
                    "or intentions",
           "index_in_sae": 31935,
           "max_activation_strength": 1}
        ]
      },
      "value": 0.5
    }
  ],
  "scopes": [],
  "name": "controller__48491998",
  "nonzero_strength_threshold": None
}
self.model.parameters["controller"] = controller
```

A.4.4 Experimental Sweep

We sweep offer amounts from 10 to 90 tokens:

```
run_ultimatum_experiments(
    baseline_agents=dummy_agents,
    fairness_agents=fairness_agents,
    model=model,
    start=10, end=90, step=5
)
```

For each offer amount we run three experiments:

- BASELINE (neutral persona, no controller)
- PROMPTING (altruistic persona, no controller)
- STEERING (neutral persona, altruism feature boosted)

Each run writes one CSV per condition. A summary table tracks:

- `accept_percent` = fraction of agents choosing "Accept"
- We then plot the three curves across offer amounts

What we log.

- CSV per condition and offer: `ultimatum_baseline_{offer}.csv`, `ultimatum_prompting_{offer}.csv`, `ultimatum_steering_{offer}.csv`
- Feature activations file: `feature_activations.txt`
- Summary CSV: `ultimatum_experiment_summary.csv`

A.5 Capability Task Implementation and Evaluation

For capability tasks (divergent creativity and product innovation), we use the same three-scenario structure (Baseline, Prompting, Steering) but add a fourth scenario with high temperature sampling to test whether randomness enhances creative outputs.

A.5.1 Task Implementation

We implement two capability tasks inspired by the Torrance Tests of Creative Thinking:

- **Divergent Creativity**: "List very detailed ways you can use a brick."
- **Product Innovation**: "List as many specific enhancements as you can that would make a stapler better."

Unlike preference tasks which use `QuestionMultipleChoice`, capability tasks use open-ended text generation to allow agents to produce diverse, creative responses.

A.5.2 Creativity Evaluation

To quantitatively assess the creativity of agent responses, we employ a secondary LLM-based evaluation using `QuestionLinearScale`. The evaluation prompts another instance of the model to rate responses on a 1 to 10 scale based on four established dimensions of creative thinking:

```
eval_question = QuestionLinearScale(
  question_name="creativity_score",
  question_text=f"""Rate the creativity of the
following task and its response on a scale of
1-10, where:

The task is: {task_descriptions[task]}

Factors to consider:

1. Fluency: The ability to produce a significant
   number of relevant ideas in response to a
   given question.
2. Flexibility: The variety of categories from
   which one can generate ideas. It's the ability
   to think of alternatives, shift from one class
   or perspective to another, and to approach a
   given problem or task from different angles.
3. Originality: The uniqueness of the ideas
   generated. Original ideas are those that are
   rare or unconventional, differing from the
   norm.
4. Elaboration: The ability to expand upon,
   refine, and embellish an idea. It involves
   adding details, developing nuances, and
   building upon a basic concept to make it more
   intricate or complex.

Response to evaluate:
{response_text}

Rate the creativity (you can be generous):""",
  question_options=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
  option_labels={
    1: "Not creative (generic, obvious, "
       "conventional ideas)",
    10: "Very creative (highly original, novel, "
        "unconventional, diverse ideas)"
  }
)
```

This multi-dimensional evaluation framework aligns with established creativity research (Torrance 1966; Guilford 1967) and allows us to systematically compare how prompting, steering, and temperature manipulation affect creative output quality.

A.5.3 High Temperature Scenario

For capability tasks only, we add a fourth scenario testing the hypothesis that increased sampling randomness correlates with creative output. We set temperature to a higher value (e.g., 0.9 or 1.0) while keeping all other parameters constant. This scenario helps disentangle whether creativity improvements from SAE steering are simply due to increased output variance or reflect genuine mechanistic changes in idea generation.

A.6 Feature Activation Analysis

We can inspect activated features for any prompt/response pair and write a compact text report:

- Directory: `safe_risky_choice/results_{YYYYMMDD_HHMMSS}/`
- File: `feature_activations.txt`
- Contents: per response, the user prompt, model response, and top-k features with activation values

Implementation sketch (as in `get_feature_activations`):

```
inspect_variant = base_variant \
    if use_base_variant else self.variant
inspector = client.feature.inspect(
    messages=[
      {"role":"system","content":"..."},
      {"role":"user",
       "content": user_prompt_text},
      {"role":"assistant",
       "content": model_response_text},
    ],
    model=inspect_variant
)
top = inspector.top(k=10)  # (feature, activation)
```

We call this for Baseline, Prompting, and Steering. For baseline we set `use_base_variant=True` to avoid any stale controller state.

A.7 Experimental Sweep and Outputs

We sweep risky-outcome magnitudes:

```
run_reward_experiments(
    baseline_agents=dummy_agents,
    slightly_risk_agents=slightly_risk_agents,
    model=model,
    start=10, end=180, step=5,
    analyze_activations=True  # toggled depending on runtime budget
)
```

For each reward we instantiate and run three experiments:

- BASELINE (neutral persona, no controller)
- SLIGHTLY PROMPTING (risk-leaning persona, no controller)
- STEERING (neutral persona, controller attached)

Each run writes one CSV per condition. A summary table is aggregated in-memory to compute:

- `percentage_risky` = fraction of agents choosing "Risky Option"
- We then plot the three curves across reward magnitudes

The helper `plot_safe_risky_choices()` automatically picks the most recent `results_*` folder and renders a comparison plot of the three conditions.

A.8 Invariants and Controls

- **Identical decoding and scaffolding across conditions.** Only the persona text (Prompting) or controller (Steering) differ. Temperatures/top-p are held constant within a sweep. Caching is disabled where we collect activations.
- **Schema enforcement.** EDSL validators ensure answers use valid MCQ labels, and rationales are present. One standardized reprompt on violation.
- **No leakage.** The Prompting trait never mentions payoff tables or numeric thresholds. Steering retains the exact prompts as Baseline.
- **Stability checks.** We prefer "lite" nudges when strong values saturate. We also check that small weight perturbations do not flip decisions arbitrarily in obviously unstable regimes.

A.9 Reproducibility and Artifacts

Every run logs:

- Timestamped results directory
- CSVs per condition and reward
- Controller JSON (if steering) with feature UUIDs and weights

- `feature_activations.txt` with top-k features per response
- Prompt columns we rely on for inspection:
  - `prompt.safe_risky_choice_user_prompt`
  - `generated_tokens.safe_risky_choice_generated_tokens`
  - `answer.safe_risky_choice`
  - `comment.safe_risky_choice_comment`

B Technical Appendix for Probe-based Steering and Analysis

B.1 Stack Overview

Software stack. We implement probe-based steering in PyTorch using HuggingFace Transformers for model inference, and scikit-learn for training ℓ_2-regularized logistic regression probes.

Base model and representation. Probe-based steering uses a frozen base language model and a linear probe trained on intermediate activations. In our main experiments we use Qwen-2-7B-Instruct and select a single layer for intervention based on held-out probe performance (layer 17 in our Qwen runs).

Activation extraction. To compute probe scores, we run a forward pass and capture the hidden state tensor at the selected layer using a forward hook. We use the final token representation as a compact summary of the prompt context.

```
# Sketch of probe score extraction
layer = model.model.layers[layer_idx]
acts = []

def capture_hook(module, input, output):
    hidden = output[0] if isinstance(output, tuple) else output
    acts.append(hidden.detach())
    return output

handle = layer.register_forward_hook(capture_hook)
_ = model(**tokenizer(prompt, return_tensors="pt"))
handle.remove()

last_token = acts[0][0, -1, :].float()
score = torch.dot(last_token, probe_direction_unit)
```

Probe training. We fit an ℓ_2-regularized logistic regression probe to predict binary behavioral labels from activations. We standardize activations prior to training and select the regularization strength via cross-validation. We train probes across layers and choose the layer with best held-out discriminative performance. This yields both the steering direction (the probe weight vector) and the intervention site (the best layer).

B.2 Agents, Cohorts, and Trial Structure

We implement the appendix cohorts from the main paper in a probe-specific form by treating each agent as an independent decoding instance indexed by a deterministic random seed.

Baseline trials. We sample n agents per task parameter with $\lambda = 0$, producing unsteered psychometric curves and probe scores.

Steering trials. We repeat the same prompts and decoding settings but enable a single-layer probe steering hook with a specified $(\lambda,$ direction$)$.

Deterministic seeding. For preference tasks, we set the seed as a function of the task parameter index and agent ID so that trial sets are reproducible.

```
# Sketch of trial loop and deterministic seeding
for param_idx, param in enumerate(task_parameters):
  for agent_id in range(n_agents):
    seed = seed_base + param_idx * 1000 + agent_id
    torch.manual_seed(seed)
    if torch.cuda.is_available():
      torch.cuda.manual_seed_all(seed)
    prompt = make_prompt(param)
    response = generate(prompt, lambda_value, direction)
```

B.3 Tasks, Labels, and Targets

Lottery. The prompt describes a choice between a safe option (guaranteed 50 tokens, fixed across trials) and a risky option (50% chance of a variable reward, 50% chance of 0 tokens). The risky reward is varied to produce a psychometric curve. We define the target behavior as the switching point, the risky reward at which $P(\text{Risky}) = 0.5$, computed by linear interpolation across reward levels.

Ultimatum. The prompt describes an offer to split 100 tokens, where the model must answer Accept or Reject. The offer amount is varied to produce an acceptance curve. We define the target behavior as the acceptance threshold, the offer size at which $P(\text{Accept}) = 0.5$.

Binary labels for probe training. For preference tasks, we convert each trial into a binary label by median-splitting the target metric (switching point or acceptance threshold) across calibration runs. This yields balanced high-versus-low labels suitable for logistic regression.

B.4 Scenarios and How We Implement Them

Probe experiments use a fixed prompt template and vary only the steering strength and direction.

B.4.1 Baseline

Meaning. Measure unsteered behavior and probe scores under the same decoding settings used for steering.

Implementation.

- Steering strength: $\lambda = 0$

- Direction: unused
- Decoding parameters (including temperature) held fixed

What we log.

- Task parameter (risky reward or offer amount)
- Parsed choice (Safe/Risky or Accept/Reject)
- Probe score computed on the prompt context

B.4.2 Steering

Meaning. Modify the model's internal state at inference time by adding a scaled probe direction vector to hidden states at a single layer.

Intervention hook. During decoding we register a forward hook on the selected layer and add an intervention of the form $\Delta h = s\,\lambda\,v$, where v is the probe direction scaled to match typical activation magnitudes, λ is the steering strength, and $s \in \{+1, -1\}$ encodes the increase versus decrease direction. We apply the intervention only to the final portion of the sequence (the last 20% of tokens) and scale the intervention linearly from 0.5 to 1.0 over these target positions.

```
# Sketch of probe steering at one layer
sign = 1.0 if direction == "increase" else -1.0
start_idx = seq_len - max(1, seq_len // 5)
position_scale = torch.linspace(0.5, 1.0, seq_len - start_idx)
hidden[:, start_idx:, :] += (sign * lambda_value * v_scaled) * \
    position_scale.view(1, -1, 1)
```

Probe scores under steering. For analysis, we report probe scores both before and after steering. The post-steering score is computed by re-running the probe score extraction on the concatenated prompt and generated response.

B.5 Lambda Calibration and Dose-Response

To achieve a desired target behavior (for example, a specified switching point), we solve for the steering strength λ using a two-stage search. Figure 10 shows the dose-response relationship between target switching points and the required λ values for both lottery and ultimatum games.

We first evaluate a coarse grid of λ values to identify a promising region, then run a bracketed binary search until the achieved metric is within a fixed tolerance. We cache trial logs for each $(\lambda, \text{direction})$ evaluation to make repeated searches efficient.

B.6 Internal Probe Dynamics

Figure 11 analyzes the internal probe dynamics during steering. The probe activation score (y-axis) is a scalar readout computed from the model's internal hidden-state vector at the selected layer using our trained logistic-regression probe. Concretely, for each trial we capture the hidden state $h \in R^d$

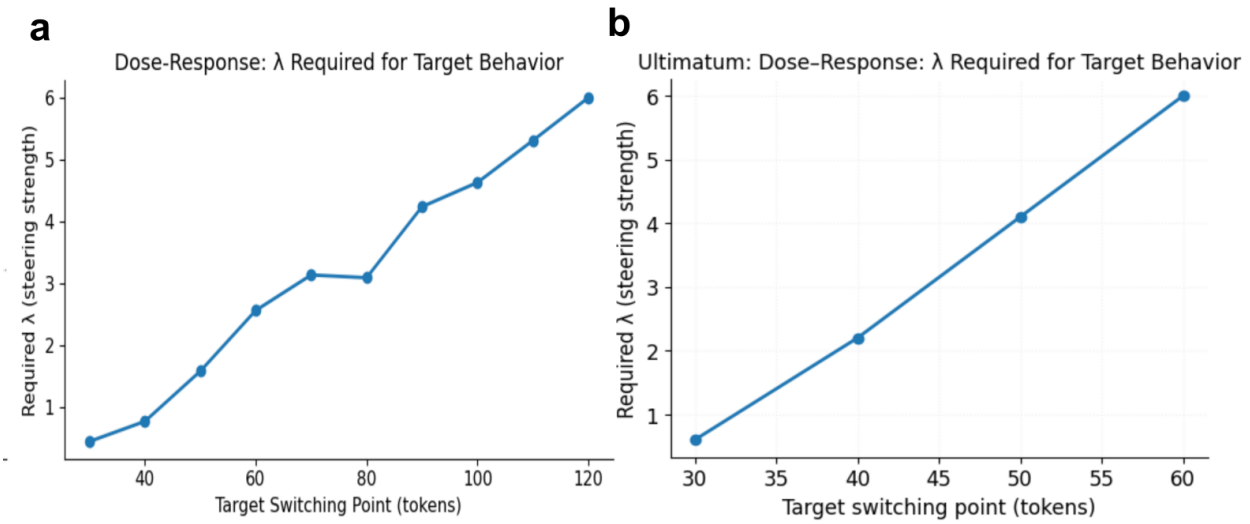


Figure 10: (a) & (b) Dose-response showing λ required to reach target switching points for lottery and ultimatum games.

at the intervention layer (we use the final prompt-token representation as a compact summary of the prompt context) and compute its signed projection onto the unit-normalized probe direction $\hat{w} = w/||w||_2$: $s = \hat{w}^\top h$ (i.e., a dot product / projection onto the probe direction). Because the probe is a logistic-regression classifier, this dot product corresponds to the probe's linear decision score up to an additive intercept term, so larger s indicates stronger alignment with the probe's positive-class direction. The distributions show that probe scores maintain clear separation between risky and safe choices across different steering strengths, validating that our intervention preserves the probe's discriminative signal rather than simply overwhelming it. Steering shifts the mean activation levels for both choice types while maintaining their relative ordering, suggesting that the probe direction captures genuine preference-related variation rather than arbitrary activation patterns. Having established that steering can shift preference-related mechanisms while preserving internal discriminability, we next ask whether the same intervention paradigm extends to capability tasks.

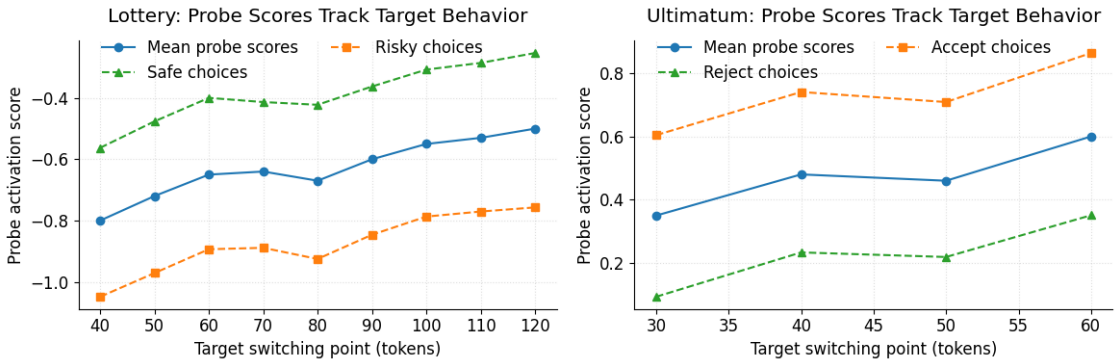
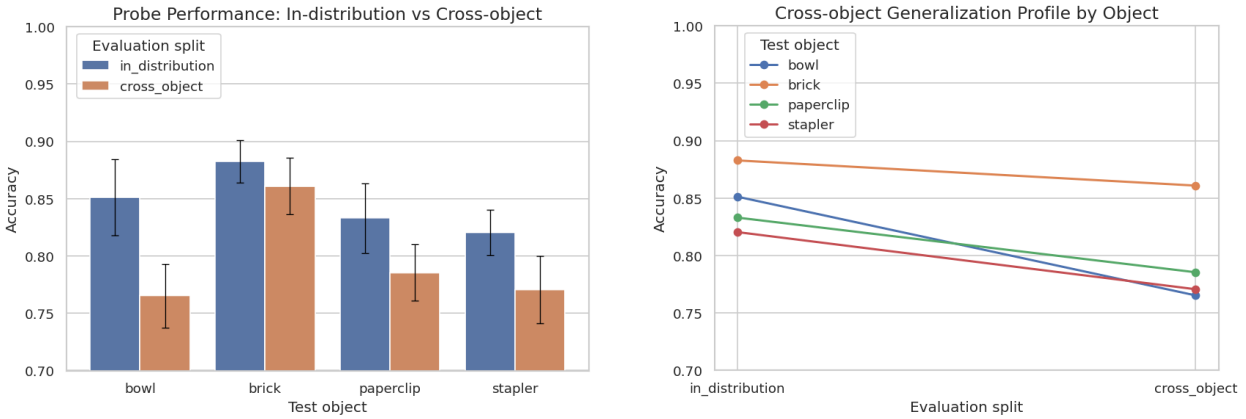


Figure 11: Probe activation score distributions for risky versus safe choices across different steering strengths.



(a) Cross-object generalization performance comparing in-distribution and cross-object probe accuracy across test objects.

(b) Performance drop from in-distribution to cross-object evaluation for each test object, showing generalization degradation patterns.

Figure 12: Cross-object generalization analysis.

B.7 Cross-Object Generalization

To assess whether learned probe directions generalize beyond their training distribution, we implement cross-object generalization experiments. We first train a logistic regression probe on hidden activations from Qwen-2-7B-Instruct for a specific construct, such as risk preference or creativity. The probe takes a single representation layer as input and predicts a binary label (risky versus safe, or high versus low creativity). For in-distribution evaluation, the probe is trained and tested on examples that all use the same object (for instance, train and test on brick). For cross-object evaluation, the probe is trained on a set of objects and tested on a held-out object that never appears in the training data (for instance, train on brick, stapler, and paperclip and test on bowl). Performance is reported as accuracy averaged over multiple random seeds.

Figure 12a compares in-distribution and cross-object performance for each test object. The results show that probes maintain reasonable accuracy when tested on held-out objects, though performance consistently drops relative to in-distribution evaluation. Figure 12b illustrates the magnitude of this performance drop for each object. The degradation ranges from 5 to 15 percentage points depending on the object, suggesting that while probe directions capture some generalizable structure, they also encode object-specific patterns that do not transfer perfectly across contexts.

B.8 Invariants and Controls

- **Identical prompts and decoding across conditions.** Within a sweep, Baseline and Steering use the same prompt templates and decoding parameters. Only λ and direction differ.
- **No prompt leakage of targets.** Prompts do not mention switching points, acceptance thresholds, or desired behaviors. Target values are only used in offline metric computation and λ selection.
- **Layer and direction selection.** We select the intervention layer by held-out probe performance and keep it fixed across all steering runs. Steering uses the probe weight vector as the

single intervention direction.

- **Seeded sampling.** We set deterministic seeds as a function of parameter index and agent ID so that trial sets are reproducible across reruns.

B.9 Reproducibility and Artifacts

Every probe run produces:

- A probe artifact (pickle) containing the probe weights, bias, layer scores, and selected best layer
- JSONL trial logs with fields including task type, task parameter, λ, direction, decoded response, parsed choice, and probe scores
- Cached trial evaluations used by the λ solver (one cache file per evaluated $(\lambda, \text{direction})$)
- Figure scripts that load cached JSONL logs to produce psychometric curves and dose-response plots