Online Supplementary Appendix

Simulating Strategic Interactions with AI Agents

July 9, 2026

Supplementary Appendix A: Review of Computational and Human-Subject Experiments in SMJ

In this Appendix, we summarize the methodology used to identify experimental studies published in the *Strategic Management Journal* (SMJ), including both agent-based models (ABMs) and human subject experiments. We then describe how these studies were classified by research topic and source of variation. While not intended to be exhaustive, this review seeks to provide a systematic overview of the questions that strategy scholars explore and the methodological tools they currently employ to identify causal mechanisms.

A.1 Compiling the List of Studies

We obtained the list of studies from Scopus, an academic database curated by Elsevier. Specifically, we used Scopus' advanced query functionality to search titles, abstracts, and author-supplied keywords for all articles published in SMJ. Our first query targeted experimental studies using human subjects, including both field and laboratory experiments:

> TITLE-ABS-KEY("field experiment") OR TITLE-ABS-KEY("RCT") OR TITLE-ABS-KEY("randomized controlled trial") OR TITLE-ABS-KEY("randomised controlled trial") OR TITLE-ABS-KEY("lab") OR TITLE-ABS-KEY("laboratory") OR TITLE-ABS-KEY("controlled experiment") AND SRC-TITLE("Strategic Management Journal") AND LIMIT-TO(EXACTSRCTITLE, "Strategic Management Journal")

This search returned 44 articles. After manually filtering out papers that were not actually experimental (e.g., special issue introductions or methodological pieces), we retained 31 studies for

analysis. Our second query aimed to identify studies using computational experiments, including agent-based models, system dynamics, and other simulation-based methods:

TITLE-ABS-KEY("agent based model") OR TITLE-ABS-KEY("computer simulation") OR TITLE-ABS-KEY("simulated experiment") OR TITLE-ABS-KEY("simulation") OR TITLE-ABS-KEY("computational simulation") OR TITLE-ABS-KEY("computational model") OR TITLE-ABS-KEY("nk model") OR TITLE-ABS-KEY("nk fitness") OR TITLE-ABS-KEY("system dynamics") OR TITLE-ABS-KEY("genetic algorithms") OR TITLE-ABS-KEY("cellular automata") OR TITLE-ABS-KEY("nk landscape") OR TITLE-ABS-KEY("fitness landscape") OR TITLE- ABS-KEY("abm") AND SRCTITLE("Strategic Management Journal") AND LIMIT-TO(EXACTSRCTITLE, "Strategic Management Journal")

This search returned 83 articles. After manually excluding studies that did not use computational modeling techniques, we retained 49 studies. The final list of studies is shown below in Appendix Table A1.

Table A1: List of SMJ Studies Included in the Systematic Review

Simulations	**Experiments**
Srikanth and Ungureanu, 2025	Novelli and Spina, 2024
Sharapov and Ross, 2023	Santamaria et al., 2024
Ketkar and Workiewicz, 2022	Hurst et al., 2024
Martignoni and Keil, 2021	Boulongne et al., 2024
Dong, 2021	Healey and Hodgkinson, 2024
Davis and Aggarwal, 2020	Wang et al., 2023
Posen et al., 2020	Kotha et al., 2023
Chen et al., 2019	Dahlander et al., 2023
Posen and Martignoni, 2018	Richter et al., 2023
Kim and Anand, 2018	Mickeler et al., 2023
Posen et al., 2018	Durand and Huysentruyt, 2022
Luoma et al., 2017	Nai et al., 2022
Aggarwal et al., 2017	Lane et al., 2021
Csaszar and Levinthal, 2016	Tong et al., 2021
Lee and Puranam, 2016	Billinger et al., 2021
Rahmandad and Repenning, 2016	Hasan and Koning, 2019
Lee et al., 2016	Chatterji et al., 2019
Martignoni et al., 2016	Flammer and Kacperczyk, 2019
Andersen and Bettis, 2015	Meier et al., 2019
Keyhani et al., 2015	Laureiro-Martínez and Brusoni, 2018
Miller and Lin, 2015	Christensen et al., 2017
Sakhartov and Folta, 2015	Elfenbein et al., 2017
Baumann and Stieglitz, 2014	Chatterji et al., 2016
Fang et al., 2014	Døjbak et al., 2016
Baum et al., 2014	Phadnis et al., 2015
Kogut et al., 2014	Jones et al., 2015
Schilling and Fang, 2014	Cain et al., 2015
Knudsen et al., 2014	Di et al., 2014
Garcia-Sanchez et al., 2014	Robert et al., 2011
Posen et al., 2013	Knez and Camerer, 1994
Markle, 2011	Brockner et al., 1993
Hu et al., 2011	
Aggarwal et al., 2011	
Miller et al., 2009	
Andersen et al., 2007	
Gavetti et al., 2005	
Gary, 2005	
Miller and Arikan, 2004	
Knott, 2003	
Zott, 2003	
Johnson and Hoopes, 2003	
Crossland and Smith, 2002	
Lee et al., 2002	
Adner, 2002	
Levy, 1994	
Mezias and Glynn, 1993	
Istvan, 1992	
Merten, 1991	
Morecroft, 1984	

A.2 Classifying Studies by Topic

Next, after reviewing the studies listed above, we inductively derived eight distinct topic categories based on clusters of related papers. While we acknowledge that this is inherently a subjective exercise and that some overlap between categories is inevitable, it nonetheless highlights meaningful groupings of research activity within the literature. In total, we derived eight distinct topic categories:

1. Competitive Strategy: How firms compete and why some perform better than others.

2. Corporate Strategy: How firms create value within the organization, including decisions about resource allocation, diversification, and strategy implementation.

3. Adaptation & Environmental Change: How firms adjust their strategies and positioning in response to changes in the external environment, be it technological, competitive, or due to other shifts.

4. Cognition & Decision-Making: How behavioral microfoundations, such as perception, cognition, and emotion, influence strategic decision-making.

5. Strategic Human Capital: How firms attract, develop, and manage talent to create value.

6. Entrepreneurship: How new ventures are created, and how individuals generate value, innovate, and disrupt industries.

7. Organizational Knowledge: How knowledge is accumulated, applied, and transferred within and between organizations.

8. Networks: How the structure of connections among individuals, organizations, or institutions shapes strategy.

We then manually classified each of the studies in Appendix Table A1 into one of the eight topic categories based on the author-supplied keywords and titles, and in ambiguous cases the

abstracts or introductions. To demonstrate how these classifications were applied, we provide illustrative examples in Appendix Table A2.

Table A2: Example Studies by Topic Category

Topic	Example
Competitive Strategy	*Time delays, competitive interdependence, and firm performance* (Luoma et al., 2017)
Corporate Strategy	*Implementation strategy and performance outcomes in related diversification* (Gary, 2005)
Adaptation & Environmental Change	*Adaptive capacity to technological change: A microfoundational approach* (Aggarwal et al., 2017)
Cognition & Decision Making	*Mental representation and the discovery of new strategies* (Csaszar and Levinthal, 2016)
Strategic Human Capital	*The Janus face of artificial intelligence feedback: Deployment versus disclosure effects on employee performance* (Tong et al., 2021)
Entrepreneurship	*Demand pull versus resource push training approaches to entrepreneurship: A field experiment* (Santamaria et al., 2024)
Organizational Knowledge	*Engineering serendipity: When does knowledge sharing lead to knowledge production?* (Lane et al., 2021)
Networks	*Does evidence of network effects on firm performance in pooled cross-section support prescriptions for network strategy?* (Baum et al., 2014)

A.3 A Taxonomy of the Experimental Sources of Variation

Finally, we review the full set of studies to identify which sources of variation each one leverages. In simulation papers, these sources often take the form of alternative initial conditions, model parameters, or exogenous shocks, whereas in experimental studies, they often correspond to the randomized interventions introduced by researchers. Observing recurring themes across papers, we inductively developed a set of categories to classify these sources of variation. Note that not every study will experimentally manipulate all of them.

The first three categories relate to the *agents* studied, such as individual decision-makers or organizations.

1. **Agent Objectives:** What agents are trying to achieve. This includes preferences (e.g., utility functions or risk tolerance), specific performance aspirations, and the nature of their goals (e.g., maximizing profit or promoting social welfare).
2. **Agent Capabilities:** What agents have at their disposal to achieve their objectives. This includes any resource endowments, internal abilities (e.g., the capacity to explore a larger search space), and private knowledge about the environment.
3. **Agent Decision Rules:** How agents pursue their objectives, leveraging their capabilities. This includes decision-making rules, heuristics, routines, algorithms (e.g., greedy search or soft-max), and other behavioral policies.

The remaining six categories relate to the environments in which agents operate.

1. **Population:** The number of agents active in the environment.
2. **Time Horizon:** The temporal structure of the environment, such as the number of periods or rounds.
3. **Choice Landscape:** The nature of the choices available to agents. For example, in NK models, this corresponds to N (the number of choice variables) and K (the degree of interdependence among them).
4. **Payoff Structure:** The rewards associated with each choice. This can simply be the magnitudes of earnings, or it can include complex features such as probabilistic payoffs or payoff interdependence across agents' choices.
5. **Information Set:** The degree to which relevant aspects of the environment are public knowledge for the agents.
6. **Spatial Structure:** The topological arrangement of the landscape (e.g., the correlation in payoff across similar choices), which could be geographical or network-based

For each study, we record whether any variation is employed for each given category. We provide an illustrative example for one such study in the Appendix Table A3 below.

Table A3: Coding the Sources of Variation for Csaszar and Levinthal (2016)

Dimension	**Coded Value**	**Explanation**
Agent Objectives	0	Managers' only objective is to maximize performance metrics
Agent Capabilities	1	They vary the strength of managers' cognitive limit (M')
Agent Decision Rules	1	They vary the search strategy (e.g., exploration parameter ε_p)
Population	0	The population is held fixed
Time Horizon	1	They show how search changes over 2000 periods
Choice Landscape	1	They vary the interdependence of choices (K)
Payoff Structure	1	They vary the contribution of rewards to profits (w)
Information Set	0	The information set is held fixed
Spatial Structure	0	There are no topological features

Supplementary Appendix B: Current Applications of LLMs in Research

Large Language Models (LLMs) have captured public attention for their impressive performance across a variety of domains (Bubeck et al., 2023), including image generation (Qu et al., 2023) and computer programming (Kazemitabaar et al., 2024). Increasingly, LLMs are being used as tools for academic research, with their versatility proving to be useful for various tasks (Grimes et al., 2023). We list the main use cases of LLMs in research in the Appendix Table B1.

Table B1: Outline of How LLMs are Being Used in Academic Research

Application	**Description**	**Example(s)**
Dataset Cleaning	Preprocess and clean datasets	Chong, Hong, and Manning (2022)
Data Analysis	Run statistical methods, including regressions	Ma et al. (2023)
Data Annotation	Annotate textual data and create labels mapping into theoretical constructs	Carlson and Burbano (2026)
Data Visualization	Create visual representations of data	Ye et al. (2024)
Scientific Writing	Refine language, including drafting full abstracts and manuscripts	Salvagno, Taccone, and Gerli (2023)
Literature Reviews	Automate search for articles	Agarwal et al. (2024)
Summarization	Generates accessible summaries of scientific content	Anderson et al. (2023)
Refining Research Instruments	Design and refine tools like survey questionnaires	Grimes et al. (2023); Charness, Jabarian, and List (2023)
Survey Respondents	Use LLMs as survey respondents	Argyle et al. (2023); Brand, Israeli, and Ngwe (2023); Li et al. (2024)
Replicate Experiments	Replicate known behavioral experiments, like the Milgram shock experiment	Aher, Arriaga, and Kalai (2023); Ashokkumar et al. (2024), Horton (2023)
Hypothesis Generation	Ask LLMs to formulate new testable hypotheses or research ideas	Doshi and Hauser (2024); Si, Yang, and Hashimoto (2024)

Several uses of LLMs leverage their advanced capabilities with data management. Researchers have leveraged LLMs to clean datasets (Chong et al., 2022), perform data analysis (Ma et al., 2023), annotate textual data (Carlson and Burbano, 2025), and generate visualizations of key variables and their relationships (Ye et al., 2024). Beyond data tasks, LLMs have been used extensively in scientific writing, even for generating abstracts and drafts. For instance, it is estimated that up to 20% of the content in computer science conference papers is now substantially AI-modified (Liang et al., 2024). Among other uses, LLMs have been leveraged to conduct systematic literature reviews (Agarwal et al., 2024) and to refine research instruments, such as survey questionnaires (Grimes et al., 2023; Charness et al., 2023).

Besides these applications, one feature of LLMs that may prove particularly useful for academic research is their remarkable tendency to exhibit human-like traits. For example, a recent study by Mei et al. (2024) subjected LLM chatbots to a personality test and a series of behavioral games, finding that their responses were statistically indistinguishable from randomly picked human subjects. Another body of research finds that LLMs rely on human-like heuristics and are prone to similar cognitive errors, reasoning, and even moral judgment (Dillion et al., 2023; Lampinen et al., 2024; Hagendorff, 2024). Given these similarities, a growing number of researchers are seeking to understand whether there are insights we can learn about humans just by indirectly studying the behavior of LLMs (Bail, 2024; Grossmann et al., 2023). In this context, LLMs would not be inputs into the research process, but rather the subjects of study, serving as proxies for humans (Manning et al., 2024; Park et al., 2023). While LLMs do not perfectly mirror human decision-making (Tjuatja et al., 2023; Mohammadi, 2024), they might provide a flawed but potentially very useful approximation as "homo silicus" (Horton, 2023).

Indeed, an active body of literature is developing around this promise. Scholars have tried to use LLMs as survey respondents to uncover consumer demand (Brand et al., 2023; Li et al., 2024) and predict voting behavior (Argyle et al., 2023). However, most research to date has focused on replicating lab-based experiments using LLMs in the place of human subjects. Aher et al. (2023) and Ashokkumar et al. (2024) replicate a suite of behavioral experiments, from the wisdom of crowds to the Milgram shock experiment, finding results consistent with past studies using humans. Horton (2023) investigates the strategic capabilities of LLMs by having them participate in games like the prisoner's dilemma or the dictator game. While these games involve two players only, more recent studies have incorporated more players and richer design elements, like information deficiencies and spatial reasoning (Wu et al., 2023; Xu et al., 2023).

Finally, some studies are using LLMs for research ideation and direct hypothesis generation. Doshi and Hauser (2024) find that when fiction writers use generative AI to obtain ideas for a story, the narratives are evaluated as more novel but tend to be more similar to each other. Similar results are found by Si et al. (2024) when tasking an LLM with generating new research ideas and comparing them with human experts. Manning et al. (2024) study automated hypothesis generation employing LLMs to propose potential causal relationships and design experiments to test those relationships.

Supplementary Appendix C: Additional Information on EDSL

C.1 What is EDSL?

Expected Parrot Domain-Specific Language (EDSL) is an open-source Python library designed to streamline and automate AI-powered research workflows, particularly those requiring large-scale coordination of LLMs (Horton and Horton, 2024). At its core, EDSL enables structured interactions with the language models: each model receives a scenario-specific, parameterized question, and their response is returned as a formatted result. We use this structure to replicate the laboratory experiment described in Hoelzemann et al. (2025). Since all interactions relied on APIs, EDSL supplied the necessary guardrails to format inputs and parse outputs reliably. Beyond our use case, EDSL is also tailored for applications such as:

- Designing and administering surveys and experiments
- Performing data labeling and content moderation
- Conducting computational social science and market research
- Gathering and analyzing responses from both humans and AI systems

C.2 Example EDSL Script

We used multiple script variations to simulate different configurations of the experiment, depending on the specific assumptions employed (e.g., payoff ambiguity and rivalry). Below, we provide an example for the specific case of non-rivalry and non-ambiguity. Note that many aspects of the environment are parameters manipulated by the researcher, whose precise value will depend on the specific simulation.

Instructions

General Information:

Welcome. This is an experiment in the economics of decision-making. If you pay close attention to these instructions, you can earn a significant amount of money paid to you at the end of the experiment. Following these instructions, you will be asked to make some choices. There are no correct choices. Your choices depend on your preferences and beliefs, so different participants will usually make different choices. You will be paid according to your choices, so read these instructions carefully and think before you decide.

The Basic Idea:

There are `{len(MOUNTAINS_DISTRIBUTION)}` mountains and each of them hides one type of gem, which can only be found by exploring the mountain. There are 3 types of gems hidden in the `{len(MOUNTAINS_DISTRIBUTION)}` mountains: Diamonds, Rubies, and Topazes. The exact values of the topazes, rubies, and diamonds vary across rounds but the diamonds are always worth more than the rubies and the rubies are always worth more than the topazes. You choose which mountains to explore and the value of the gems you find are your earnings in dollars. Your objective is to maximize your own earnings.

Location of Gems:

`{location_of_gems}`

The Mapped Mountain:

At the beginning of each round, one mountain will be randomly selected to be mapped and its gem value will be revealed to all participants. Each participant will be able to see the same gem contained by the mountain. The mountain chosen for mapping is random and changes in each round. Besides the value of the mapped mountain, no participant has any other initial information in Stage 1 on the location of gems.

How Participants Choose Mountains:

There are `{no_of_players}` participants including you in total. In each round, participants choose which mountain to explore. The choice does not happen simultaneously, but participants choose sequentially, one after the other, according to a random order. You can choose to explore any mountain you wish or select the mapped mountain. If you choose the same mountain chosen by other participants, each of you will receive the full value uncovered. Similarly, if someone else chooses the same mountain that you previously chose, you will still receive the full gem's value (and so will the other participant(s) who chose it). This means that payoffs are non-rival and there is no penalty in choosing the same mountain as other players. To repeat, no participant has any private information in Stage 1 on the location of the gems, besides the common knowledge about the mapped mountain.

Each Round Has 2 Stages:

A round consists of 2 stages. At the beginning of a new round, gems are randomly allocated to the `{len(MOUNTAINS_DISTRIBUTION)}` different mountains. The position of gems

will not be reset between the two stages in a round. Then, before Stage 1 begins, one mountain will be mapped and its value revealed to everyone. In Stage 1, all participants sequentially choose one mountain to explore. Before choosing a mountain, you will see which mountains have been selected by the other participants in your group who chose before you. You can choose the same mountain chosen by other participants or a different mountain. At the end of Stage 1, the gems hidden in each mountain selected by all participants in Stage 1 are revealed, and you earn the value of the gem hidden in the mountain you chose. In Stage 2, you can again choose any of the same `{len(MOUNTAINS_DISTRIBUTION)}` mountains; that is, you can either choose the same mountain from Stage 1 or switch to another one. The position of gems remains the same as in Stage 1, but this time you will also see the gems located in the mountains revealed in Stage 1 in addition to the mapped mountain. At the end of Stage 2, the gems hidden in each mountain selected by all participants in Stage 2 are revealed, and you earn the value of the gem hidden in the mountain you chose in Stage 2. Your total earnings for the round equal the sum of the value of the gem you found in Stage 1 and the value of the gem you found in Stage 2. Again, if multiple players choose the same mountain, they all receive its full value.

Payment:

At the end of the round, you will be paid an amount equivalent to the sum of payoffs you earned in Stage 1 and Stage 2. This protocol of determining payments suggests that you should choose in each Stage knowing that your choice directly determines your payment because the dollar value of the gems you select will directly translate into your earnings.

Frequently Asked Questions:

- Q1: Is this some kind of psychology experiment with an agenda you haven't told us?
  A: No, it is an economics experiment. These instructions are meant to clarify how you earn money and our interest is in seeing how people make decisions.

- Q2: Is there a "correct" or "wrong" choice of action? Is this kind of a test?
  A: No, your optimal choice depends on your preferences and beliefs and different people may hold different beliefs.

- Q3: Will there be any mountains with empty payoffs (no gem at all)?
  A: No, each and every mountain contains a gem and you are guaranteed a payoff for choosing any mountain whether it is a topaz, ruby, or a diamond.

- Q4: Will there be any negative payoffs for some hidden mountains?
  A: No, there is no payoff lower than that of a topaz, which is always positive.

- Q5: Are the payoffs split if more players choose the same mountain?
  A: No, each participant receives the full value of the gem uncovered, regardless of how many participants choose a mountain in that specific stage.

Instructions for the Risk Preference:

The Choice: You will be asked to choose between two options, "Option A" and "Option B" where:

"Option A" always pays $4.00 with probability p and $3.20 otherwise.

"Option B" always pays $7.70 with probability p and $0.20 otherwise.

Repeated Choices:

You will be asked to make a choice between "Option A" and "Option B" not once, but ten times where p will increase from 10% to 100%, 10% at a time.

For example, the first choice will have p=10% and you will choose whether you prefer "Option A" ($4.00 with a 10% chance or $3.20 otherwise) or "Option B" ($7.70 with a 10% chance or $0.20 otherwise).

Each successive choice will increase p by 10 percentage points until the last choice where "Option A" will pay $4.00 with certainty, and "Option B" will pay $7.70 with certainty.

Note: Once you switch from choosing "Option A" to "Option B", it makes sense that you will continue to choose "Option B" in all consecutive choice problems. For example, if you prefer "Option B" when p=80%, then it makes sense to prefer "Option B" when p=90% and when p=100%, since "Option B" is even more attractive in these choice problems.

Payment for risk preference task:

The computer will randomly select one of the 10 choice problems and pay you according to your choice in that problem where the computer will decide the outcome based on the value of p.

C.3 Link to Code Repository

In the interest of transparency and open science, we provide the full code used to simulate the experiments. The implementation is available in a public GitHub repository. We include two Jupyter notebooks corresponding to different information conditions in the game:

- **No-data condition:** Participants make decisions without observing the value of any alternative.

  https://github.com/arulmabr/theorizing_with_llms/blob/main/notebooks/streetlight_no-data.ipynb

- **Data condition:** Participants observe the value of one alternative at the outset, either low, medium, or high in value.
  https://github.com/arulmabr/theorizing_with_llms/blob/main/notebooks/streetlight_with-data.ipynb

Supplementary Appendix D: Key Parameters for LLM-Powered Experiments

Environment: The researcher must specify the common set of conditions that the agent(s) encounters during an in silico experiment. The following environmental dimensions are especially relevant to strategic management:

1. **Population:** The number of agents active in the environment. The computational environment theoretically supports an unlimited number of agents.
2. **Time Horizon:** The temporal structure of the environment, such as the number of periods or rounds. The computational environment theoretically supports an unlimited number of rounds.
3. **Choice Landscape:** The nature of the choices available to agents. The computational environment theoretically supports an unlimited number of choices, or combinations of choices, enabling the tunable interaction complexity seen in NK models (Kauffman, 1993; Levinthal, 1997).
4. **Payoff Structure:** The rewards associated with each choice. This can simply be the magnitudes of earnings, or can include complex features like probabilistic payoffs or payoff rivalry across agents. The computational environment theoretically supports unlimited combinations of earning schedules.
5. **Information Set:** The degree to which relevant aspects of the environment are public knowledge. The computational environment makes this straightforwardly tunable, which can be done in the initial instructions. For instance, agents may be told the options available, but not the payoffs associated with each option.

6. **Spatial Features:** The topological arrangement of the landscape, which could be geographical or network-based. The computational environment theoretically supports an unlimited number of spatial arrangements of agents and choices.

Agents: The researcher must also specify each agent's characteristics. The following agentic dimensions are especially relevant to strategic management:

1. **Objectives:** What agents are trying to achieve. This includes preferences (e.g., risk tolerance), specific performance aspirations, and the nature of their goals - such as maximizing profit or promoting social welfare. LLMs can be tuned to pursue different objectives by prompting them with additional scripts. They can also be assigned specific roles, which might contain implicit objectives that the language model needs to infer. For example, a "student LLM" might focus on maximizing learning, while a "firm LLM" might choose to maximize profits.
2. **Capabilities:** What agents have at their disposal. This includes any resource endowments, internal abilities (e.g., the capacity to explore a larger search space), and private knowledge about the environment. Once the agent is informed of these capabilities, they must be encoded throughout the implementation to ensure the engine accurately tracks and constrains the agent's behavior.
3. **Decision-Rules:** How agents pursue their goals. This includes decision-making rules, heuristics, algorithms (e.g., greedy search, softmax), and other behavioral policies. This once more entails prompting them with additional scripts.

Turn-taking Procedures: The researcher must specify how decisions are made by agents. There are two primary methodologies they can choose from:

1. **Sequential Decision-Making:** Agents are allotted a "turn" to make their decision. After each turn, the state space is updated and the experiment proceeds until a predetermined endpoint has been reached. The researcher specifies how turns are allotted, e.g., using a random order or perhaps having an independent LLM interact with each agent, and then decide (Manning et al., 2024).

2. **Simultaneous Decision-Making:** Agents take actions at the same time within a round. After each round, the state space is updated, and the experiment proceeds until a predetermined endpoint has been reached. The researcher would simply need to decide the number of rounds (Huang et al., 2024).

Communication Channels: The researcher must decide whether the AI agents can communicate with each other during the experiment (or engage in other forms of interaction, such as trading). This functionality must be programmed into the engine, but remains feasible with LLMs (Xu et al., 2023).

Outcomes: The researcher must specify which outcomes to record as a function of the research question and the theoretical constructs explored.

1. **Quantitative Data:** This includes participant choices, their earnings as a function of collective decisions, the time or effort taken to make a decision, or any other measurable outcome. The engine can also calculate more complex corollary measures. This data could then be analyzed using traditional econometric techniques for rigorous inference.
2. **Qualitative Responses:** This primarily includes the explanations that AI agents provide for their decisions, but could include anything in light of LLMs' language comprehension and generation capabilities (e.g., their subjective experiences navigating a certain

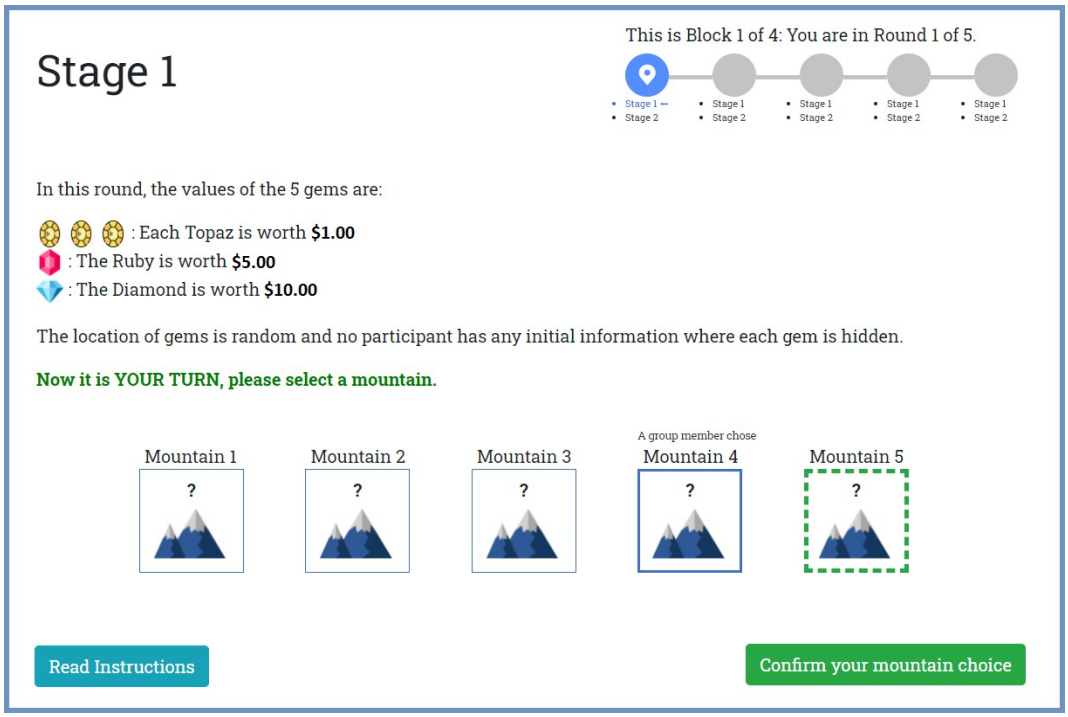
environment). Collecting these responses is essential for verifying that the experiments are working as intended (e.g., seeing whether an agent is embodying an assigned role).

Interventions: These are isolated changes to any element of the game environment or the agents themselves. With LLMs, it becomes possible to pull levers that are not normally available in real-world lab experiments, such as manipulating an agent's preferences or tolerance for risk. Once all the building blocks have been specified, the researcher can begin simulating the experiment, adjusting one element at a time and tracing how the outcomes of interest evolve.

Supplementary Appendix E: Additional Figures and Tables

Figure E1: The Experimental Platform from Hoelzemann et al. (2025)

Panel A: User interface

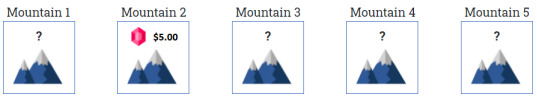


Panel B: Examples of no-data condition and data conditions

(i) No-data condition

(ii) Low-value condition

(iii) Medium-value condition

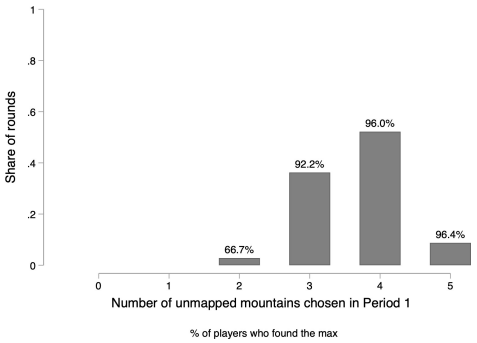


(iv) High-value condition

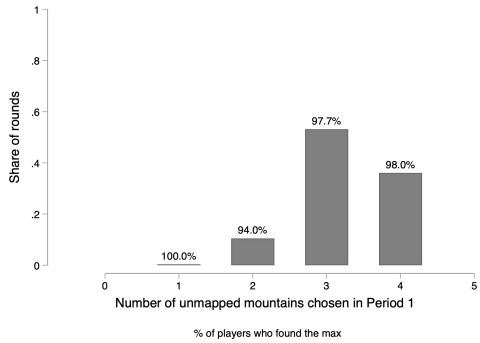
Note: This figure depicts the software platform used in the online lab experiment of Hoelzemann et al. (2025). Panel A shows how the interface is seen by participants in the no-data condition. The values of the gems for each round are shown in the upper left corner. In this example, the user can see that Mountain 4 has already been picked and decides to select Mountain 5. Panel B depicts the four experimental conditions. For instance, in plot (iii), the mountain uncovering the ruby is revealed at the start of the round. The figure is reproduced with permission from Hoelzemann et al. (2025).

Figure E2: Number of Unknown Mountains Chosen in Period 1 by Human Subjects

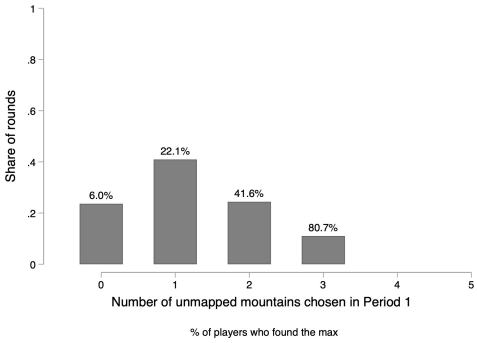
(i) No-data condition



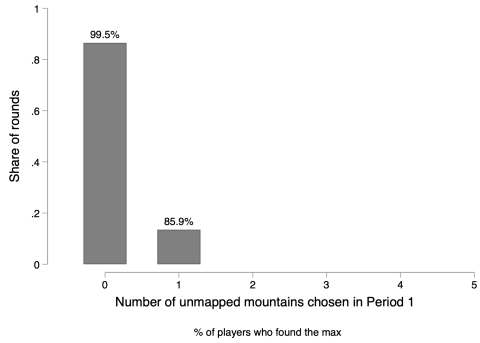
(ii) Low-value condition



(iii) Medium-value condition

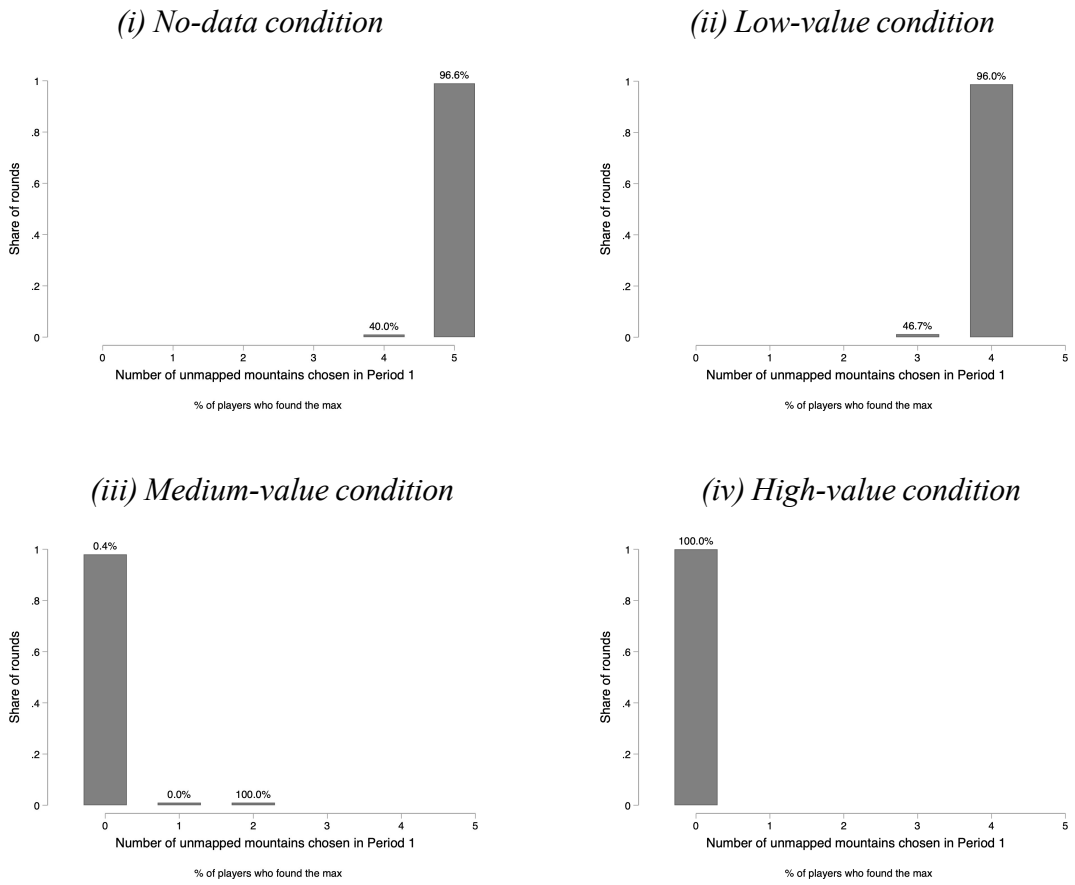


(iv) High-value condition



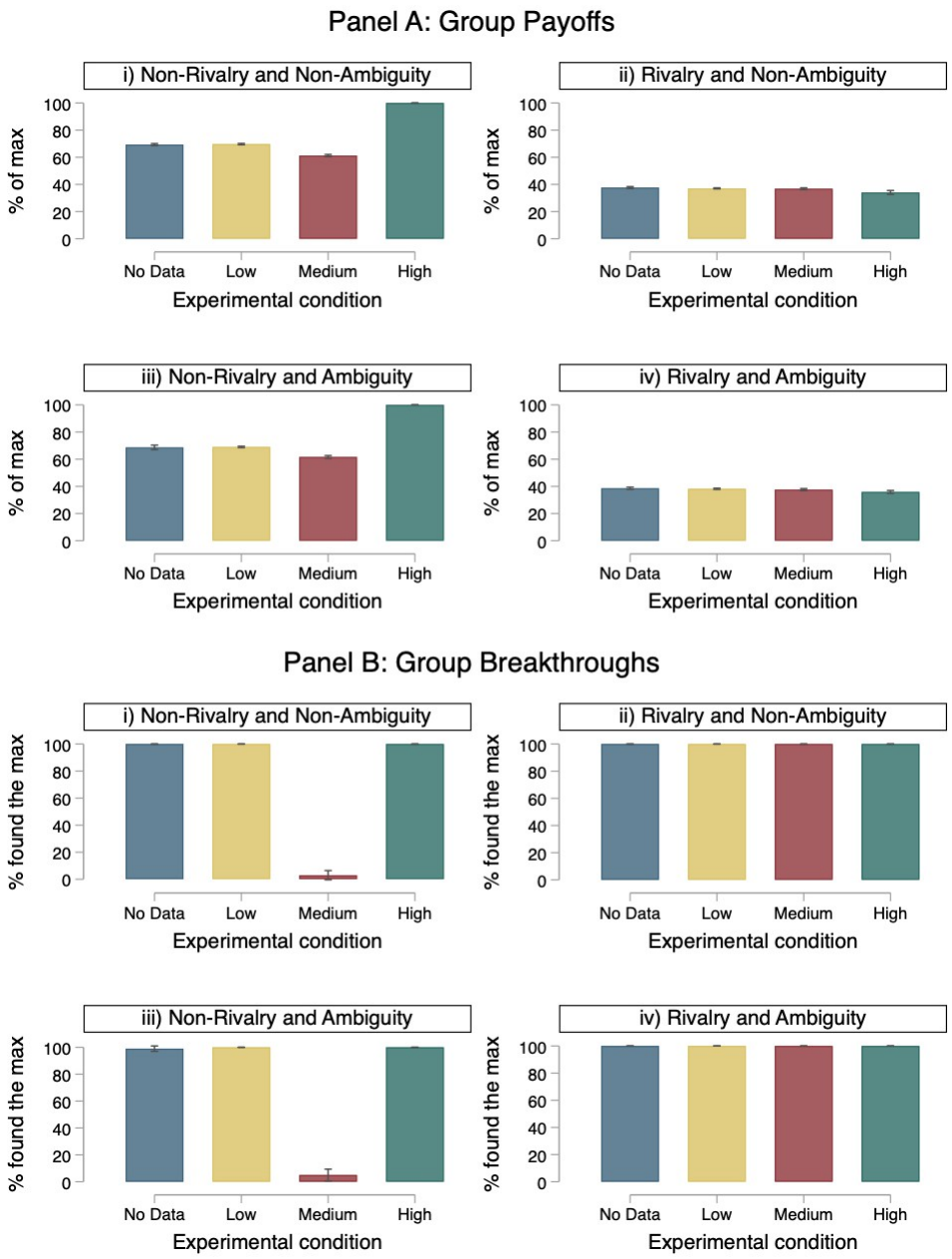
Note: Each plot represents the empirical frequencies of rounds for each possible number of unknown options chosen in period 1, shown separately by experimental condition. The text above each bar shows the share of participants who found the maximum payoff in period 2. The figure is reproduced with permission from Hoelzemann et al. (2025).

Figure E3: Number of Unknown Mountains Chosen in Period 1 by AI Agents (Baseline)



Note: Each plot represents the empirical frequencies of rounds for each possible number of unknown options chosen in period 1, shown separately by experimental condition. The text above each bar shows the share of participants who found the maximum payoff in period 2.

Figure E4: Relaxing Theoretical Assumptions in LLM-based Simulations



Note: This figure shows how the outcomes of the original experiment change when we relax key theoretical assumptions using LLMs. In Panel A, the outcome of interest is the average group earnings (as a percentage of the maximum possible earnings). In Panel B, the outcome of interest is the mean likelihood of a breakthrough, defined as at least one group member finding the diamond. Plot i) shows the baseline results, where there is no rivalry or ambiguity in payoffs. In plot ii), we introduce rivalry in payoffs, which means agents who choose the same gem have to split the payoff. In plot iii), we introduce ambiguity in payoffs, which means the distribution of gems is not known from the outset. In plot iv), we introduce both rivalry and ambiguity in payoffs.

Table E1: Quantitative Results of Experiment with Human Subjects

Panel A: Round-level Outcomes

	Individual payoff	I(Individual found max)	I(Group found max)
	(1)	(2)	(3)
High	6.682***	0.039**	0.003
	(0.143)	(0.011)	(0.006)
Low	0.889***	0.037**	0.006
	(0.096)	(0.011)	(0.007)
Medium	−2.261***	−0.645***	−0.539***
	(0.214)	(0.028)	(0.039)
Constant	13.349***	0.968***	1.012***
	(0.163)	(0.018)	(0.020)
Round order FE	Yes	Yes	No
Block order FE	Yes	Yes	Yes
Payoff structure FE	Yes	Yes	Yes
Observations	7000	7000	1400

Panel B: Analysis of Mechanisms

	Exploration	Individual payoff		I(Individual found max)		I(Group found max)	
	Round	Period 1	Period 2	Period 1	Period 2	Period 1	Period 2
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
High	-75.059***	6.499***	0.191*	0.784***	0.037**	0.310***	0.003
	(1.770)	(0.097)	(0.073)	(0.008)	(0.011)	(0.015)	(0.006)
Low	5.744***	0.664***	0.225**	0.055***	0.036**	0.122***	0.006
	(1.300)	(0.074)	(0.062)	(0.007)	(0.011)	(0.020)	(0.007)
Medium	−34.130***	1.251***	−3.511***	−0.134***	−0.644***	−0.444***	−0.539***
	(2.450)	(0.122)	(0.142)	(0.007)	(0.027)	(0.029)	(0.039)
Constant	83.977***	3.610***	9.717***	0.187***	0.963***	0.752***	1.012***
	(2.045)	(0.104)	(0.117)	(0.008)	(0.018)	(0.018)	(0.020)
Round Order FE	No	Yes	Yes	Yes	Yes	No	No
Block order FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Payoff structure FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Observations	1400	7000	7000	7000	7000	1400	1400

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Standard errors clustered at the session level in parentheses. Estimates from OLS models. In Panel A, the sample in Columns 1 and 2 is at the participant-round level (5 participants × 1400 rounds). The sample in Column 3 is at the group-round level (1400 rounds). *Individual payoff* = participant-level round payoffs in Canadian dollars; *I(Individual found max)*:0/1=1 if the location of the maximum was found by the participant; *I(Group found max)*:0/1=1 if the location of the maximum was found by at least one participant in the round. The excluded category captured by the constant is the condition without data. In Panel B, the sample in Column 1 is at the group-round level (1400 rounds). The sample in Columns 2, 3, 4, and 5 is at the participant-period level (5 participants × 1400 periods of each type). The sample in Columns 6 and 7 is at the group-period level (1400 periods of each type). *Exploration*= share of unknown mountains explored in the round; *Individual payoff* = participant-level period payoffs in Canadian dollars; *I(Individual found max)*:0/1=1 if the location of the maximum was found by the participant in the period; *I(Group found max)*:0/1=1 if the location of the maximum was found by at least one participant in the period. The table is reproduced with permission from Hoelzemann et al. (2025).

Table E2: Quantitative Results of Experiment with AI Agents (Baseline)

Panel A: Round-level Outcomes

	Individual payoff	I(Individual found max)	I(Group found max)
	(1)	(2)	(3)
High	6.777***	0.022***	−0.001
	(0.183)	(0.007)	(0.002)
Low	0.067	−0.011	0.000
	(0.207)	(0.008)	(0.002)
Medium	−1.805***	−0.964***	−0.969***
	(0.182)	(0.008)	(0.017)
Constant	15.423***	0.978***	1.000***
	(0.176)	(0.007)	(0.001)
Round order FE	Yes	Yes	No
Payoff structure FE	Yes	Yes	Yes
Observations	2500	2500	500

Panel B: Analysis of Mechanisms

	Exploration	Individual payoff		I(Individual found max)		I(Group found max)	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
	Round	Period 1	Period 2	Period 1	Period 2	Period 1	Period 2
High	−100.000***	6.569***	0.208***	0.802***	0.040***	0.010	0.000
	(0.085)	(0.183)	(0.049)	(0.018)	(0.009)	(0.010)	(0.002)
Low	−0.100	0.077	−0.009	0.007	−0.006	0.002	0.000
	(0.115)	(0.205)	(0.057)	(0.021)	(0.011)	(0.012)	(0.001)
Medium	−98.250***	2.258***	−4.063***	−0.196***	−0.946***	−0.980***	−0.970***
	(0.806)	(0.179)	(0.060)	(0.018)	(0.010)	(0.014)	(0.017)
Constant	100.000***	4.531***	10.892***	0.198***	0.960***	0.990***	1.000***
	(0.049)	(0.176)	(0.049)	(0.018)	(0.009)	(0.010)	(0.001)
Round order FE	No	Yes	Yes	Yes	Yes	No	No
Payoff structure FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Observations	500	2500	2500	2500	2500	500	500

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Standard errors clustered at the session level in parentheses. Estimates from OLS models. In Panel A, the sample in Columns 1 and 2 is at the participant-round level (5 participants × 500 rounds). The sample in Column 3 is at the group-round level (500 rounds). *Individual payoff* = participant-level round payoffs in Canadian dollars; *I(Individual found max)*:0/1=1 if the location of the maximum was found by the participant; *I(Group found max)*:0/1=1 if the location of the maximum was found by at least one participant in the round. The excluded category captured by the constant is the condition without data. See text for more details. In Panel B, the sample in Column 1 is at the group-round level (500 rounds). The sample in Columns 2, 3, 4, and 5 is at the participant-period level (5 participants × 500 periods of each type). The sample in Columns 6 and 7 is at the group-period level (500 periods of each type). *Exploration*= share of unknown mountains explored in the round; *Individual payoff* = participant-level period payoffs in Canadian dollars; *I(Individual found max)*:0/1=1 if the location of the maximum was found by the participant in the period; *I(Group found max)*:0/1=1 if the location of the maximum was found by at least one participant in the period.

Table E3: Summary of Results Using LLMs as Synthetic Subjects

Intervention	Description	Rounds (#)	Cost ($)	Hypothetical payment ($)	Summary of findings
Baseline	Original experiment with 5 players and 5 mountains.	500	$250	$50,000	Players herd around the suboptimal option, reducing overall payoffs.
Extension 1: Varying the experimental parameters					
Varying group size	Group size varies from 5 to 30 agents.	150	$300	$60,000	No change in exploration: social conformity reinforces herding around the suboptimal option.
Varying the choice landscape	The number of mountains varies from 5 to 30.	150	$75	$15,000	No change in exploration: agents are not swayed by a larger absolute number of options.
Varying payoff magnitude	Gem values are amplified one millionfold.	500	$250	N/A	No change in exploration: the theory depends on relative, rather than absolute, payoff magnitudes.
Extension 2: Relaxing theoretical assumptions					
Relaxing payoff rivalry	Agents choosing the same gem must split its payoff.	500	$250	$50,000	Rivalry breaks the herding effect because agents seek to avoid splitting payoffs.
Relaxing payoff ambiguity	Agents are no longer told how many gems of each kind exist.	500	$250	$50,000	No change in exploration: the theory does not depend on prior information about the payoff distribution.
Relaxing payoff rivalry and ambiguity	Agents must split payoffs and are not told the underlying distribution.	500	$250	$50,000	Rivalry dominates ambiguity and continues to break the herding effect.
Extension 3: Manipulating agent preferences					
Incorporating risk seeking	The number of risk-loving agents varies from 1 to all 5 group members.	500	$250	$50,000	Risk seeking breaks the herding effect: exploration rises with the number of risk-loving agents.
Incorporating prosociality	The number of prosocial agents varies from 1 to all 5 group members.	250	$125	$25,000	Prosociality breaks the herding effect: exploration rises with the number of prosocial agents.
Incorporating explorativeness	The number of explorative agents varies from 1 to all 5 group members.	250	$125	$25,000	Explorativeness breaks the herding effect: exploration rises with the number of explorative agents.
Total		**3,800**	**$2,125**	**$375,000**	**Excludes hypothetical payments from the amplified-payoff case.**

Note: This table provides an overview of all the results from this study. Column 1 lists the extensions we introduce. Column 2 provides a description of each extension. Column 3 indicates the number of rounds simulated. Column 4 shows the approximate total cost in GPT-4 credits as of July 2024. Column 5 highlights the amount we would have needed to pay to human subjects. Column 6 summarizes the key result. Row 2 covers the baseline replication of the original lab experiment. Rows 4-6 cover changes in the experimental setup. Rows 8-10 cover key theoretical assumptions being relaxed. Rows 12-14 cover heterogeneous agent preferences being introduced.

Table E4: Comparing LLM-based Simulations to Direct Elicitation

				Mean group payoff (%)			Mean group breakthrough (%)		
Configuration	**Data**	**Rivalry**	**Ambiguity**	**Humans**	**AI Agents**	**Prediction**	**Humans**	**AI Agents**	**Prediction**
1	None	FALSE	FALSE	68	69	75	99	100	80
2	Low			72	70	62	100	100	80
3	Medium			58	61	75	46	3	80
4	High			98	100	95	100	100	100
5	None	TRUE	FALSE	-	38	45	-	100	40
6	Low			-	37	62	-	100	80
7	Medium			-	37	75	-	100	40
8	High			-	34	75	-	100	100
9	None	FALSE	TRUE	-	69	50	-	99	20
10	Low			-	69	62	-	100	40
11	Medium			-	62	62	-	5	40
12	High			-	100	75	-	100	100

Note: This table shows the results when we try asking GPT-4 to predict the outcomes of the experiments. Column 1 lists the specific prediction task. Column 2 lists the data condition. Columns 3 and 4 list which theoretical assumptions apply. Column 7 shows GPT-4's direct prediction for the average group earnings (as a percentage of the maximum possible earnings). This is compared with outcomes derived from human subjects (Column 5) and AI agents (Column 6). Similarly, Column 10 shows GPT-4's direct prediction for the mean likelihood of a breakthrough, which occurs when at least one group member finds the diamond. This is compared with outcomes derived from human subjects (Column 8) and AI agents (Column 9). In Rows 1-4, we show the predictions for the baseline replication. In Rows 5-8, we present the predictions for the extension that introduces payoff rivalry. In Rows 9-12, we present the predictions for the extension that introduces payoff ambiguity.

Supplementary Appendix F: Full Simulation Code and Parameters

This appendix reports the full details of our LLM-powered simulations, in compliance with SMJ's GenAI use policy.

1. Sampling generation settings:

For the simulations, we used OpenAI's GPT-4-Turbo through EDSL. In the code/notebooks, the model is instantiated as Model('gpt-4-turbo'). We did not manually override the sampling parameters. Based on the EDSL/OpenAI defaults, the parameters are:

temperature=0.5, top_p=1, max_tokens=1000, frequency_penalty=0, presence_penalty=0, logprobs=False, top_logprobs=3; one completion requested per agent decision; no fixed LLM seed set.

The simulation also used Python randomization for gem locations and agent order. So non-determinism came from both the LLM generation process and the simulation engine's random assignment/order. Before simulation batches, we cleared the EDSL cache so that decisions were generated from fresh model calls rather than retrieved from cached responses.

2. Number of runs per condition:

- Baseline replication: 500 total (No data: 100; Low data: 250; Medium data: 100; High data: 50)
- Parameter sweeps: 800 total
    - Group size: 150 total (10 agents: 50; 20 agents: 50; 30 agents: 50; All in the medium-data condition.)

    - Choice landscape: 150 total (10 mountains: 50; 20 mountains: 50; 30 mountains: 50; All in the medium-data condition.)
    - Magnified payoffs: 500 total (No data: 100; Low data: 250; Medium data: 100; High data: 50)
- Assumption relaxations: 1,500 total
    - Rivalry only: 500 total (No data: 100; Low data: 250; Medium data: 100; High data: 50)
    - Ambiguity only: 500 total (No data: 100; Low data: 250; Medium data: 100; High data: 50)
    - Rivalry + ambiguity: 500 total (No data: 100; Low data: 250; Medium data: 100; High data: 50)
- Preference manipulations: 1,000 total
    - Risk-seeking agents: 500 total (1 risk-seeking agent: 100; 2 risk-seeking agents: 100; 3 risk-seeking agents: 100; 4 risk-seeking agents: 100; 5 risk-seeking agents: 100)
    - Pro-social agents: 250 total (1 pro-social agent: 50; 2 pro-social agents: 50; 3 pro-social agents: 50; 4 pro-social agents: 50; 5 pro-social agents: 50)
    - Explorative agents: 250 total (1 explorative agent: 50; 2 explorative agents: 50; 3 explorative agents: 50; 4 explorative agents: 50; 5 explorative agents: 50)

3. Repository with simulation scripts:

The public code is here: https://github.com/arulmabr/theorizing_with_llms/tree/main/notebooks

The repository includes the no-data and data-condition simulation notebooks and the prompt structure.

4. Full simulation code:

We report the original notebook implementation. Model and generation settings used in the reported simulations:

- model: OpenAI `gpt-4-turbo` via EDSL
- EDSL/OpenAI defaults: temperature=0.5, top_p=1, max_tokens=1000, frequency_penalty=0, presence_penalty=0, logprobs=False, top_logprobs=3
- one completion requested per agent decision
- no fixed LLM seed set

The simulation engine uses Python randomization for gem locations and agent choice order. If a model call fails or returns an unparseable answer, the code falls back to a random valid mountain choice and records `error_flag = 1`.

```
========================================

from __future__ import annotations

import os
import random
import re
from dataclasses import dataclass, asdict
from pathlib import Path
from typing import Dict, Iterable, List, Optional, Sequence, Tuple

import pandas as pd
from edsl import Agent, Cache, Model, Survey
from edsl.questions import QuestionMultipleChoice

# ----------------------------------------------------------------------
```

```
# Model and reported sampling configuration
# ------------------------------------------------------------------------

MODEL_NAME = "gpt-4-turbo"

# These are the EDSL/OpenAI default generation parameters to report.
The
# original notebooks instantiate Model('gpt-4-turbo') and do not
manually
# override these parameters.
REPORTED_DEFAULT_MODEL_PARAMETERS = {
    "temperature": 0.5,
    "max_tokens": 1000,
    "top_p": 1,
    "frequency_penalty": 0,
    "presence_penalty": 0,
    "logprobs": False,
    "top_logprobs": 3,
}


def make_model() -> Model:
    """Create the EDSL model object used in the experiments."""
    model = Model(MODEL_NAME)
    # The original notebooks set rate limits for throughput only. This
does not
    # change sampling behavior.
    model._set_rate_limits(rpm=10000, tpm=2000000)
    return model


# ------------------------------------------------------------------------
# Experimental parameters
# ------------------------------------------------------------------------

MOUNTAINS_DISTRIBUTION = ["low", "low", "low", "medium", "high"]
GEMS_MAPPING = {"low": "Topaz", "medium": "Ruby", "high": "Diamond"}
DEFAULT_PAYOFFS = [
    {"low": 1, "medium": 6, "high": 10},
    {"low": 1, "medium": 6, "high": 10.5},
    {"low": 1, "medium": 7, "high": 12},
    {"low": 2, "medium": 7, "high": 11},
    {"low": 3, "medium": 8, "high": 12},
]

# Run counts used for reporting in the GenAI-policy appendix.
FOUR_CONDITION_RUNS = {"no-data": 100, "low": 250, "medium": 100,
"high": 50}
RUN_COUNTS_BY_ANALYSIS = {
    "baseline_replication": FOUR_CONDITION_RUNS,
    "magnified_payoffs": FOUR_CONDITION_RUNS,
    "rivalry_only": FOUR_CONDITION_RUNS,
```

```
    "ambiguity_only": FOUR_CONDITION_RUNS,
    "rivalry_and_ambiguity": FOUR_CONDITION_RUNS,
    "group_size": {"10_agents": 50, "20_agents": 50, "30_agents": 50},
    "choice_landscape": {"10_mountains": 50, "20_mountains": 50,
"30_mountains": 50},
    "risk_seeking_agents": {
        "1_risk_agent": 100,
        "2_risk_agents": 100,
        "3_risk_agents": 100,
        "4_risk_agents": 100,
        "5_risk_agents": 100,
    },
    "pro_social_agents": {
        "1_pro_social_agent": 50,
        "2_pro_social_agents": 50,
        "3_pro_social_agents": 50,
        "4_pro_social_agents": 50,
        "5_pro_social_agents": 50,
    },
    "explorative_agents": {
        "1_explorative_agent": 50,
        "2_explorative_agents": 50,
        "3_explorative_agents": 50,
        "4_explorative_agents": 50,
        "5_explorative_agents": 50,
    },
}


AGENT_NAMES = [
    "Jack",
    "Jane",
    "David",
    "Cobb",
    "Samantha",
    "Michael",
    "Sarah",
    "Daniel",
    "Emily",
    "Joshua",
]
AGENT_AGES = [22, 23, 24, 25, 26, 27, 28, 29, 30, 31]
AGENT_LOCATIONS = [
    "New York",
    "California",
    "Texas",
    "Florida",
    "Washington",
    "Nevada",
    "Oregon",
    "Colorado",
    "Arizona",
```

```
    "Illinois",
]


@dataclass
class SimulationConfig:
    """Parameters for one simulation batch."""

    no_of_players: int = 5
    ambiguity: bool = False
    rivalrous: bool = False
    distribution: Sequence[str] = tuple(MOUNTAINS_DISTRIBUTION)
    payoff_sets: Sequence[Dict[str, float]] = tuple(DEFAULT_PAYOFFS)
    output_path: str = "simulation_output.csv"


# ---------------------------------------------------------------------
# Prompt construction
# ---------------------------------------------------------------------


def location_of_gems_text(distribution: Sequence[str], ambiguity:
bool) -> str:
    """Return instructions about the distribution/location of gems."""
    if ambiguity:
        return (
            "Each mountain could hold a Diamond, a Ruby, or a Topaz.
However, "
            "which mountains contain which gems is unknown, and nobody
knows how "
            "many gems per type there are. At the beginning of each
round, each "
            "type of gem is randomly assigned to a different mountain,
so any gem "
            "could be hidden in any mountain."
        )
    return (
        f"In each round, there are: {distribution.count('low')}
mountain(s) "
        f"containing topazes; {distribution.count('medium')}
mountain(s) "
        f"containing rubies; {distribution.count('high')} mountain(s)
containing "
        "diamonds. However, which mountains contain which gems is
unknown. At "
        "the beginning of each round, each type of gem is randomly
assigned to "
        "a different mountain, so any gem could be hidden in any
mountain."
    )
```

```
def build_agent_instructions(config: SimulationConfig) -> str:
    """Build the experimental instructions given to each AI agent."""
    distribution = list(config.distribution)
    loc_text = location_of_gems_text(distribution, config.ambiguity)
    payoff_rule = (
        "If you choose the same mountain chosen by other participants,
each of "
        "you will equally split the value of the uncovered gem in that
specific "
        "stage. This means payoffs are rival and there is a penalty in
choosing "
        "the same mountain as other players."
        if config.rivalrous
        else
        "If you choose the same mountain chosen by other participants,
each of "
        "you will receive the full value uncovered. This means payoffs
are "
        "non-rival and there is no penalty in choosing the same
mountain as other "
        "players."
    )

    split_faq = (
        "Yes. If multiple agents choose the same mountain, they
equally split "
        "its full value in that specific stage."
        if config.rivalrous
        else
        "No. Each participant receives the full value of the gem
uncovered, "
        "regardless of how many participants choose the same mountain
in that "
        "specific stage."
    )

    return f"""
General Information:
Welcome. This is an experiment in the economics of decision-making. If
you pay close attention to these instructions, you can earn a
significant amount of money paid to you at the end of the experiment.
Following these instructions, you will be asked to make some choices.
There are no correct choices. Your choices depend on your preferences
and beliefs, so different participants will usually make different
choices. You will be paid according to your choices, so read these
instructions carefully and think before you decide.

The Basic Idea:
There are {len(distribution)} mountains and each of them hides one
type of gem, which can only be found by exploring the mountain. There
are three types of gems: Diamonds, Rubies, and Topazes. The exact
values of the topazes, rubies, and diamonds vary across rounds but
```

```
diamonds are always worth more than rubies and rubies are always worth
more than topazes. You choose which mountains to explore and the value
of the gems you find are your earnings in dollars. Your objective is
to maximize your own earnings.

Location of Gems:
{loc_text}

The Mapped Mountain:
At the beginning of each data-condition round, one mountain may be
mapped and its gem value revealed to all participants. Each
participant sees the same revealed mountain. Besides the mapped
mountain, no participant has any other initial information in Stage 1
on the location of gems.

How Participants Choose Mountains:
There are {config.no_of_players} participants including you in total.
In each round, participants choose sequentially, one after another,
according to a random order. You can choose any mountain you wish.
{payoff_rule}

Each Round Has 2 Stages:
A round consists of two stages. At the beginning of a new round, gems
are randomly allocated to the mountains. The position of gems is not
reset between the two stages. In Stage 1, all participants
sequentially choose one mountain to explore. Before choosing, you see
which mountains have been selected by previous participants in your
group. At the end of Stage 1, the gems hidden in the selected
mountains are revealed. In Stage 2, you can again choose any mountain;
you can either choose the same mountain from Stage 1 or switch. In
Stage 2, you also see the gems located in the mountains revealed in
Stage 1. Your total earnings for the round equal the sum of your Stage
1 and Stage 2 payoffs.

Frequently Asked Questions:
Q1: Is this some kind of psychology experiment with an agenda you
haven't told us?
A: No. It is an economics experiment. These instructions clarify how
you earn money.
Q2: Is there a correct or wrong choice?
A: No. Your optimal choice depends on your preferences and beliefs.
Q3: Will any mountains have empty payoffs?
A: No. Each mountain contains a gem.
Q4: Will there be negative payoffs?
A: No. The topaz is always the lowest payoff and is always positive.
Q5: Are payoffs split if more players choose the same mountain?
A: {split_faq}
""".strip()


# ---------------------------------------------------------------------
# Agent construction and choices
```

```
# ---------------------------------------------------------------------


def make_agents(config: SimulationConfig) -> List[Agent]:
    """Create AI agents with fixed demographic traits and common
instructions."""
    instructions = build_agent_instructions(config)
    agents: List[Agent] = []
    for idx in range(config.no_of_players):
        agent = Agent(
            name=AGENT_NAMES[idx % len(AGENT_NAMES)],
            traits={
                "age": AGENT_AGES[idx % len(AGENT_AGES)],
                "location": AGENT_LOCATIONS[idx %
len(AGENT_LOCATIONS)],
                "agent_id": idx + 1,
                "risk_priming": "None",
                "prosociality": "None",
                "exploration_priming": "None",
            },
            instruction=instructions,
        )
        agents.append(agent)
    return agents


def set_agent_priming(
    agents: List[Agent],
    risk_count: int = 0,
    prosocial_count: int = 0,
    explorative_count: int = 0,
) -> None:
    """Optionally prime the first k agents for preference-manipulation
extensions."""
    risk_text = (
        "This agent is risk-loving. If the diamond has not been found
yet, they "
        "will pick an unchosen mountain to give them the chance of
discovering "
        "one, since this is the kind of risk they like to take."
    )
    prosocial_text = (
        "This agent is pro-social. This means they will pick unchosen
mountains "
        "to give the other members of the group the chance of
discovering a "
        "diamond. This agent is only successful if the others choose
the diamond, "
        "since it wants the other agents to earn the highest possible
payoff."
    )
    explorative_text = (
```

```
        "This agent only wants to find the diamond. They will never
pick a ruby "
        "or a topaz, because that is not their goal. They will keep
picking "
        "unchosen mountains until they find the diamond, since this is
their "
        "ultimate goal."
    )
    for i, agent in enumerate(agents):
        agent["traits"]["risk_priming"] = risk_text if i < risk_count
else "None"
        agent["traits"]["prosociality"] = prosocial_text if i <
prosocial_count else "None"
        agent["traits"]["exploration_priming"] = explorative_text if i
< explorative_count else "None"


def mountain_options(num_mountains: int) -> List[str]:
    return [f"Mountain {i}" for i in range(1, num_mountains + 1)]


def parse_mountain_index(choice: str) -> int:
    """Convert a string like 'Mountain 3' to zero-indexed integer
2."""
    match = re.findall(r"\d+", str(choice))
    if not match:
        raise ValueError(f"Could not parse mountain choice:
{choice!r}")
    return int(match[0]) - 1


def agent_response(
    agent: Agent,
    context: str,
    model: Model,
    options: Sequence[str],
) -> Tuple[str, str, int]:
    """Prompt one agent for a mountain choice.

    Returns: (choice, comment, error_flag)
    """
    shuffled_options = list(options)
    random.shuffle(shuffled_options)
    question = QuestionMultipleChoice(
        question_name="select_mountain",
        question_text=(
            "You are being asked to choose a mountain. "
            f"Context: {context}\nWhich mountain do you choose?"
        ),
        question_options=shuffled_options,
    )
```

```
    try:
        result = question.by(agent).by(model).run(cache=Cache())
        choice = str(result.select("answer.select_mountain").first())
        comment =
str(result.select("comment.select_mountain_comment").first())
        if choice not in options:
            raise ValueError(f"Invalid option returned: {choice}")
        return choice, comment, 0
    except Exception as exc:  # match original fallback logic, but
retain error flag
        fallback_choice = random.choice(list(options))
        return (
            fallback_choice,
            f"Fallback random choice due to model/parsing error:
{type(exc).__name__}",
            1,
        )


# ---------------------------------------------------------------------
# Game simulation
# ---------------------------------------------------------------------


def compute_stage_payoffs(
    choices: Sequence[str],
    hidden_layout: Sequence[str],
    payoff_values: Dict[str, float],
    rivalrous: bool,
) -> List[float]:
    """Compute agent payoffs for one stage."""
    if not rivalrous:
        return [payoff_values[hidden_layout[parse_mountain_index(c)]]
for c in choices]
    counts = {choice: list(choices).count(choice) for choice in
set(choices)}
    return [
        payoff_values[hidden_layout[parse_mountain_index(choice)]] /
counts[choice]
        for choice in choices
    ]


def simulate_game(
    overall_round_number: int,
    sub_round_number: int,
    payoff_values: Dict[str, float],
    reveal_type: str,
    agents: List[Agent],
    model: Model,
    config: SimulationConfig,
) -> pd.DataFrame:
```

```
    """Simulate one two-stage group round and return row-level
data."""
    distribution = list(config.distribution)
    hidden_layout = random.sample(distribution, len(distribution))
    options = mountain_options(len(hidden_layout))
    inverse_payoffs = {v: k for k, v in payoff_values.items()}

    revealed_mountain = "None"
    mountain_map: Dict[str, float] = {}

    if reveal_type in payoff_values:
        for idx, gem_class in enumerate(hidden_layout):
            if gem_class == reveal_type:
                revealed_mountain = f"Mountain {idx + 1}"
                mountain_map[revealed_mountain] =
payoff_values[gem_class]
                break

    if config.ambiguity:
        context = (
            "Mountains are hidden in a random order. Payoffs are
hidden in "
            "mountains in a random order and all mountains have a gem,
with "
            "Diamond > Ruby > Topaz."
        )
    else:
        context = (
            f"This round has {distribution.count('low')} payoff(s) of
value "
            f"${payoff_values['low']} (Topazes),
{distribution.count('medium')} "
            f"payoff(s) of value ${payoff_values['medium']} (Ruby),
and "
            f"{distribution.count('high')} payoff(s) of value
${payoff_values['high']} "
            "(Diamond). Mountains are hidden in a random order."
        )

    if revealed_mountain != "None":
        gem_name = GEMS_MAPPING[reveal_type]
        context += f"\nRevealed Mountain: {revealed_mountain} has a
gem {gem_name}."
    else:
        context += "\nNo mountains are revealed for this round."

    agent_order = list(range(len(agents)))
    random.shuffle(agent_order)
    rows = []

    # Stage 1
    stage1_choices: List[str] = []
```

```
    context += "\nAgents are making their choices for Stage 1."
    for order, agent_idx in enumerate(agent_order, start=1):
        choice, comment, error_flag = agent_response(
            agents[agent_idx], context, model, options
        )
        stage1_choices.append(choice)
        mountain_idx = parse_mountain_index(choice)
        payoff_value = payoff_values[hidden_layout[mountain_idx]]
        mountain_map[choice] = payoff_value
        context += f"\nAn Agent chose {choice}."
        rows.append(
            {
                "round_number": overall_round_number,
                "condition": reveal_type,
                "risk_priming":
agents[agent_idx]["traits"].get("risk_priming", "None"),
                "ambiguity": config.ambiguity,
                "rivalrous": config.rivalrous,
                "revealed_mountain": revealed_mountain,
                "sub_round_number": sub_round_number,
                "stage_number": 1,
                "agent_id": agent_idx + 1,
                "choice": choice,
                "choice_order": order,
                "comment_by_the_agent": comment,
                "agent_payoff": 0.0,  # updated below
                "total_payoff_of_the_mountain": payoff_value,
                "payoff_class": inverse_payoffs.get(payoff_value),
                "mountains_arrangement": hidden_layout,
                "mountains_payoffs": payoff_values,
                "prosociality":
agents[agent_idx]["traits"].get("prosociality", "None"),
                "exploration_priming":
agents[agent_idx]["traits"].get("exploration_priming", "None"),
                "error_flag": error_flag,
            }
        )

    stage1_payoffs = compute_stage_payoffs(
        stage1_choices, hidden_layout, payoff_values, config.rivalrous
    )
    for row, payoff in zip(rows[-len(stage1_choices):],
stage1_payoffs):
        row["agent_payoff"] = payoff

    context += "\nLearnings from Stage 1:"
    for mtn, value in mountain_map.items():
        context += f"\n{mtn} is revealed to have a value of ${value}."

    # Stage 2
    stage2_rows_start = len(rows)
    stage2_choices: List[str] = []
```

```
    context += "\nAgents are making their choices for Stage 2."
    for order, agent_idx in enumerate(agent_order, start=1):
        choice, comment, error_flag = agent_response(
            agents[agent_idx], context, model, options
        )
        stage2_choices.append(choice)
        mountain_idx = parse_mountain_index(choice)
        payoff_value = payoff_values[hidden_layout[mountain_idx]]
        context += f"\nAn Agent chose {choice}."
        rows.append(
            {
                "round_number": overall_round_number,
                "condition": reveal_type,
                "risk_priming":
agents[agent_idx]["traits"].get("risk_priming", "None"),
                "ambiguity": config.ambiguity,
                "rivalrous": config.rivalrous,
                "revealed_mountain": revealed_mountain,
                "sub_round_number": sub_round_number,
                "stage_number": 2,
                "agent_id": agent_idx + 1,
                "choice": choice,
                "choice_order": order,
                "comment_by_the_agent": comment,
                "agent_payoff": 0.0,  # updated below
                "total_payoff_of_the_mountain": payoff_value,
                "payoff_class": inverse_payoffs.get(payoff_value),
                "mountains_arrangement": hidden_layout,
                "mountains_payoffs": payoff_values,
                "prosociality":
agents[agent_idx]["traits"].get("prosociality", "None"),
                "exploration_priming":
agents[agent_idx]["traits"].get("exploration_priming", "None"),
                "error_flag": error_flag,
            }
        )

    stage2_payoffs = compute_stage_payoffs(
        stage2_choices, hidden_layout, payoff_values, config.rivalrous
    )
    for row, payoff in zip(rows[stage2_rows_start:], stage2_payoffs):
        row["agent_payoff"] = payoff

    return pd.DataFrame(rows)


# ----------------------------------------------------------------------
# Batch runners
# ----------------------------------------------------------------------


def clear_edsl_cache(cache_dir: str = ".edsl_cache") -> None:
```

```
    """Clear EDSL cache directory before a fresh simulation batch."""
    cache_path = Path(cache_dir)
    if cache_path.exists():
        import shutil

        shutil.rmtree(cache_path)


def run_four_condition_batch(
    config: SimulationConfig,
    model: Optional[Model] = None,
    condition_rounds: Optional[Dict[str, int]] = None,
    clear_cache: bool = True,
) -> pd.DataFrame:
    """Run a four-condition batch: no-data, low, medium, high."""
    if clear_cache:
        clear_edsl_cache()
    if model is None:
        model = make_model()
    if condition_rounds is None:
        condition_rounds = FOUR_CONDITION_RUNS

    agents = make_agents(config)
    all_rows = []
    overall_round = 1
    for payoff_values in config.payoff_sets:
        # Allocate rounds across payoff schedules as evenly as
possible.
        for condition, total_rounds in condition_rounds.items():
            rounds_for_this_payoff = total_rounds //
len(config.payoff_sets)
            for sub_round in range(1, rounds_for_this_payoff + 1):
                all_rows.append(
                    simulate_game(
                        overall_round,
                        sub_round,
                        payoff_values,
                        reveal_type=condition,
                        agents=agents,
                        model=model,
                        config=config,
                    )
                )
                overall_round += 1
    data = pd.concat(all_rows, ignore_index=True)
    data.to_csv(config.output_path, index=False)
    return data


if __name__ == "__main__":
    # Example: run the baseline replication configuration.
```

```
    # Requires a valid OpenAI/Expected Parrot API setup in the
environment.
    config = SimulationConfig(
        no_of_players=5,
        ambiguity=False,
        rivalrous=False,
        output_path="baseline_four_condition_simulation.csv",
    )
    run_four_condition_batch(config)
```

Supplementary Appendix References

AGARWAL, S., I. H. LARADJI, L. CHARLIN, AND C. PAL (2024): "LitLLM: A Toolkit for Scientific Literature Review," arXiv preprint arXiv:2402.01788.

AHER, G., R. I. ARRIAGA, AND A. T. KALAI (2023): "Using Large Language Models to Simulate Multiple Humans and Replicate Human Subject Studies," arXiv preprint.

ARGYLE, L. P., E. C. BUSBY, N. FULDA, J. R. GUBLER, C. RYTTING, AND D. WINGATE (2023): "Out of one, many: Using language models to simulate human samples," Political Analysis, 31, 337–351.

ASHOKKUMAR, A., L. HEWITT, I. GHEZAE, AND R. WILLER (2024): "Predicting Results of Social Science Experiments Using Large Language Models," Working Paper.

BAIL, C. A. (2024): "Can Generative AI improve social science?" Proceedings of the National Academy of Sciences, 121.

BRAND, J., A. ISRAELI, AND D. NGWE (2023): "Using GPT for market research," Harvard Business School Marketing Unit Working Paper, No. 23-062.

BUBECK, S., V. CHANDRASEKARAN, R. ELDAN, J. GEHRKE, E. HORVITZ, E. KAMAR, P. LEE, ET AL. (2023): "Sparks of Artificial General Intelligence: Early experiments with GPT-4," arXiv preprint.

CARLSON, N. AND V. BURBANO (2026): "The Use of LLMs to Annotate Data in Management Research: Guidelines and Warnings," *Strategic Management Journal,* 47, 699–725.

CHARNESS, G., B. JABARIAN, AND J. A. LIST (2023): "Generation next: Experimentation with AI," NBER Working Paper w31679.

CHONG, D., J. HONG, AND C. D. MANNING (2022): "Detecting label errors by using pre-trained language models," arXiv preprint arXiv:2205.12702.

CSASZAR, F. A. AND D. A. LEVINTHAL (2016): "Mental representation and the discovery of new strategies," Strategic Management Journal, 37, 2031–2049.

DILLION, D., N. TANDON, Y. GU, AND K. GRAY (2023): "Can AI language models replace human participants?" Trends in Cognitive Sciences, 27, 597–600.

DOSHI, A. R. AND O. P. HAUSER (2024): "Generative AI enhances individual creativity but reduces the collective diversity of novel content," Science Advances, 10, eadn5290.

GRIMES, M., G. VON KROGH, S. FEUERRIEGEL, F. RINK, AND M. GRUBER (2023): "From scarcity to abundance: Scholars and scholarship in an age of generative artificial intelligence," Academy of Management Journal, 66, 1617–1624.

GROSSMANN, I., M. FEINBERG, D. C. PARKER, N. A. CHRISTAKIS, P. E. TETLOCK, AND W. A. CUNNINGHAM (2023): "AI and the transformation of social science research," Science, 380, 1108–1109.

HAGENDORFF, T. (2024): "Deception abilities emerged in large language models," Proceedings of the National Academy of Sciences, 121.

HOELZEMANN, J., G. MANSO, A. NAGARAJ, AND M. TRANCHERO (2025): "The streetlight effect in data-driven exploration," NBER Working Paper w32401.

HORTON, J. AND R. HORTON (2024): "EDSL: Expected Parrot Domain Specific Language for AI Powered Social Science," White Paper, Expected Parrot.

HORTON, J. J. (2023): "Large Language Models as Simulated Economic Agents: What Can We Learn from Homo Silicus?" NBER Working Paper w31122.

HUANG, J., E. J. LI, M. H. LAM, T. LIANG, W. WANG, ET AL. (2024): "How Far Are We on the Decision-Making of LLMs? Evaluating LLMs' Gaming Ability in Multi-Agent Environments," arXiv preprint.

KAUFFMAN, S. A. (1993): The Origins of Order: Self-organization and Selection in Evolution, Oxford University Press.

KAZEMITABAAR, M., R. YE, X. WANG, A. Z. HENLEY, P. DENNY, M. CRAIG, AND T. GROSSMAN (2024): "CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs," Proceedings of the CHI Conference on Human Factors in Computing Systems.

LAMPINEN, A. K., I. DASGUPTA, S. C. CHAN, H. R. SHEAHAN, A. CRESWELL, D. KUMARAN, J. L. MCCLELLAND, AND F. HILL (2024): "Language models, like humans, show content effects on reasoning tasks," PNAS nexus, 3.

LEVINTHAL, D. A. (1997): "Adaptation on rugged landscapes," Management Science, 43, 934–950.

LI, P., N. CASTELO, Z. KATONA, AND M. SARVARY (2024): "Frontiers: Determining the validity of large language models for automated perceptual analysis," Marketing Science, 43, 254–266.

LIANG, W., Y. ZHANG, Z. WU, H. LEPP, W. JI, X. ZHAO, H. CAO, S. LIU, S. HE, Z. HUANG, ET AL. (2024): "Mapping the increasing use of LLMs in scientific papers," arXiv preprint arXiv:2404.01268.

MA, P., R. DING, S. WANG, S. HAN, AND D. ZHANG (2023): "InsightPilot: An LLM-empowered automated data exploration system," Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 346–352.

MANNING, B. S., K. ZHU, AND J. J. HORTON (2024): "Automated social science: Language models as scientist and subjects," NBER Working Paper w32381.

MEI, Q., Y. XIE, W. YUAN, AND M. O. JACKSON (2024): "A Turing test of whether AI chatbots are behaviorally similar to humans," Proceedings of the National Academy of Sciences, 121.

MOHAMMADI, B. (2024): "Wait, It's All Token Noise? Always Has Been: Interpreting LLM Behavior Using Shapley Value," arXiv preprint.

PARK, J. S., J. O'BRIEN, C. J. CAI, M. R. MORRIS, P. LIANG, AND M. S. BERNSTEIN (2023): "Generative Agents: Interactive Simulacra of Human Behavior," arXiv preprint arXiv:2304.03442.

QU, L., S. WU, H. FEI, L. NIE, AND T. CHUA (2023): "LayoutLLM-T2I: Eliciting Layout Guidance from LLM for Text-to-Image Generation," Proceedings of the 31st ACM International Conference on Multimedia, 643–654.

SI, C., D. YANG, AND T. HASHIMOTO (2024): "Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers," arXiv preprint.

TJUATJA, L., V. CHEN, S. T. WU, A. TALWALKAR, AND G. NEUBIG (2023): "Do LLMs exhibit human-like response biases? A case study in survey design," arXiv preprint.

WU, Y., X. TANG, T. M. MITCHELL, AND Y. LI (2023): "Smartplay: A benchmark for LLMs as intelligent agents," arXiv preprint.

XU, Y., S. WANG, P. LI, F. LUO, X. WANG, W. LIU, AND Y. LIU (2023): "Exploring large language models for communication games: An empirical study on werewolf," arXiv preprint.

YE, Y., J. HAO, Y. HOU, Z. WANG, S. XIAO, Y. LUO, AND W. ZENG (2024): "Generative AI for visualization: State of the art and future directions," Visual Informatics.