Simulating Economic Experiments Using Large Language Models: Design and Development of a Computational Tool

by

Sohini Kar

S.B., Massachusetts Institute of Technology (2022)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

Author Sohini Kar
Department of Electrical Engineering and Computer Science
May 31, 2023

Certified by John Horton
Associate Professor

Accepted by Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Simulating Economic Experiments Using Large Language Models: Design and Development of a Computational Tool

by

Sohini Kar

Submitted to the Department of Electrical Engineering and Computer Science
on May 31, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Large language models, known for capturing the syntax, semantics, and broader representation of human behavior, can be used to simulate humans in AI-based experimentation. Because these models provide responses consistent with humans, they may be used to pilot studies or glean insights into social and economic scenarios. This research presents the development of `homo silicus`, a Python-based library for simulating economic experiments and social scenarios using AI subjects. Using the library, users can design, test, and analyze results of experiments conducted *in silico*. We replicate various classical economic research to better understand how well AI subject responses align with observed human behavior, and we test the impact of parameters such as temperature and prompt engineering to determine their influence on results. The `homo silicus` library provides researchers with a cost-effective method to iterate on projects without extensive resources or specific participant recruitment. This research contributes to advancing the field of AI-powered social simulation models and their applications in economic experimentation.

Thesis Supervisor: John Horton
Title: Associate Professor

Acknowledgments

I am extremely thankful for Professor John Horton for supporting me through my MEng and this research opportunity, whose guidance and mentorship was invaluable especially as I learned more about online labor markets and economic experimentation. I would also like to thank James Brand and Mohammed Alsobay for their feedback and recommendations in developing this project.

I would also like to thank the EECS Undergraduate Office for their support and advice through every situation. I am also thankful for all the incredible professors, mentors, and teaching staff at MIT that helped me develop and grow as a student.

Thank you to all of my friends, for the support and encouragement through every late night and difficult pset. And finally, thank you to my parents and brother — Sunistha, Barun, and Ishan — and to my grandparents for their unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

The use of large language models (LLMs) in economic and social experiments to simulate human subjects has gained increasing attention for its wide array of potential benefits. From using them in simulations of potential experiments to quickly and cheaply gain preliminary insights about results [12], to testing experiment variations across input prompts and subject demographics [4], LLMs provide numerous advantages for researchers desiring fast and inexpensive insights on potential results.

Current literature on the benefits that modern LLMs provide to the field of economics focus on three main areas: (1) experiments are cheap to run, (2) results are produced quickly even across many variations on prompts and subject endowments, and (3) these language models are replicating results that align with traditional and established experiments. As technologies such as OpenAI's ChatGPT [1] become increasingly accessible to researchers and general users, LLMs are being used as alternative means for experimenters to test preliminary ideas or to build artificial datasets [6].

Researchers have been using simulations of social situations, prompting LLMs the way they would prompt a human subject, to understand in what ways the responses parallel or replicate established human behavior. As we continue to understand how AI-based experimentation may help to understand human behavior, demand for tools

to help create, structure, and run such simulations has increased [12]. This thesis outlines the design and development of a Python-based library to create AI agents that may participate in simulated economic scenarios and experiments. Through flexible and structured modules, this library provides a framework to create experiments built by iterating across many experimental conditions on both prompts and agent backgrounds, which would otherwise take incredible amounts of time and money to run on real humans. We test this library at various stages of development by replicating popular economic experiments and understanding how factors such as prompt engineering impact the viability of the framework for novel experimentation.

1.2 Objectives

In this thesis, we present the design and development of a Python-based library (called `homo silicus`) that allows users to quickly create, run, and analyze simulated economic experiments and social scenarios. Using LLMs as computational models of humans, this library provides streamlined functions and modules to allow experimenters to efficiently outline different scenarios and AI agent endowments, which are combined and sent to available APIs for responses. The responses are then analyzed and formatted into datasets for immediate use by the user.

Our key contribution is a software tool that is flexible enough to be used to simulate various types of experiments. In this thesis, we will focus on simulating primarily economic experiments, and development of supporting modules and classes will emphasize prior research on economic experimentation. Future work may extend this approach to other types of scenarios.

The secondary objectives of this thesis include effective data analysis built into the experiment pipeline as well as support for simulating multiple agents "communicating" within the scenarios. The analysis will include both quantitative and qualitative measures, as appropriate given the experiment type. For example, for a choice-based experiment, we may automate creation of graphs that analyze the number of times each choice was chosen. In more conversation-based scenarios, we may use NLP

analysis of the text of the interactions.

At various stages of development and design, we attempt to replicate prior findings using our package, and compare the effects of various parameters on results. These parameters include both prompt structure and generation as well as model parameter tuning. This work may be used to further understand how LLM-based simulations serve as valid and viable models of human behavior, and to what extent these AI-powered social simulation models may be used to generate novel hypotheses and conduct the appropriate testing and analysis in an automated fashion.

Chapter 2

Background and Related Work

2.1 Large Language Models

The field of Natural Language Processing (NLP) has experienced significant advances in recent years, particularly with the emergence of Large Language Models (LLMs). LLMs are trained on large corpora of text to "learn" natural language and be able to predict what text should follow a given input [5, 6]. Most recently, LLMs have reached a level of sophistication allowing them to be applied across a multitude of fields, from speech recognition in healthcare to software development using Github Copilot. Significantly, LLMs have shown the capability to replicate behavior recorded and researched in economic and psychological studies [12].

OpenAI's GPT family of models and successive versions have received significant attention due to their advanced capabilities and ease of use. In this paper, we focus on GPT-3 (Generative Pre-trained Transformer 3), an autoregressive language model that utilizes a transformer architecture. A transformer model uses the mechanism of self-attention, allowing it to process an entire text at once in order to weight the significance of and give context to each part of the input [5, 7].

The input to a model is therefore a sequence of words, which is then tokenized into a sequence of tokens $\mathcal{U} = \{u_1, ..., u_n\}$ [19]. A token is a sequence of characters that can represent part or an entire word, and these are used as linguistic units; OpenAI estimates a single token is approximately four characters [1]. Each possible token has

an ID that corresponds to its frequency in the language (here, English) of its training data. After processing, the output is a sequence of guesses, which is a probability distribution for each token that denotes the likelihood that the corresponding word or token may follow the input text.

GPT-3 is both a highly powerful model and a very usable tool. OpenAI maintains Python- and Julia-based APIs, allowing developers to submit numerous prompts to any of the available models (`text-ada-001`, `textbabbage-001`, `text-curie-001`, `text-davinci-001`, `text-davinci-002`, `text-davinci-003`; of these, the `text-davinci-003` model is considered the most capable and is known as GPT-3.5). These prompts may be a single message, or a list of messages simulating a chat. The models can be tuned using various parameters in the request body itself. Finally, the user receives multiple difference responses for each submission that may be further processed and utilized as needed [1]. The responses (called completions by the OpenAI API) usually consist of a conversational response or an autocomplete-style reply to the input.

Parameters available for OpenAI models [1, 19] that are significant for this project include: (1) `temperature`, (2) `top_p`, (3) `n`, and (4) `logprobs`.

1. The `temperature` option impacts the sampling temperature and the randomness of the output, where 0 is most deterministic and 2 is most random. Specifically, a general probability distribution for outputs is given by the softmax function:

$$p(u_i) = \frac{e^{u_i}}{\sum_V^{j=1} e^{u_j}} \tag{2.1}$$

Using the temperature parameter T, the softmax function becomes:

$$p(u_i) = \frac{e^{\frac{u_j}{T}}}{\sum_V^{j=1} e^{\frac{u_j}{T}}} \tag{2.2}$$

In Equation 2.2, a lower temperature $T \approx 0$ lowers the resulting probability of outputs that are relatively smaller, and a higher temperature $T \approx 2$ balances out

the output probabilities. Using the API, the default temperature setting of 1.0 generally outputs answers that are not predetermined, and many resources claim `temperature = 0.8` is the optimal variance [18, 1]. Colloquially, temperature is thought to impact the "creativity" of an LLM.

2. An alternative to sampling with temperature is the `top_p` parameter, which has the model only consider the results of the tokens with `top_p` probability mass. Using this parameter, say `top_p = 0.2`, would mean only the tokens comprising the top 20% probability mass are considered in producing the outputs. Adjusting both this and the temperature parameters is discouraged.

3. `n` denotes the number of completions to generate for each prompt.

4. OpenAI exposes the probabilities for each token in the output completion. If the outcome sequence of tokens were $\mathcal{U} = \{u_1, ..., u_n\}$, the probability of this sequence of tokens would be:

$$P(\mathcal{U}) = p(u_1, ..., u_n) = p(u_1) \cdot ... \cdot p(u_n) \tag{2.3}$$

Because each $p(u_i) < 0$ (where $i \leq n$), the resulting $p(\mathcal{U}) << 0$ for a large sequence of output tokens. However, we may convert each probability into a log probability or `logprob`. By doing so, we get:

$$L(\mathcal{U}) = \log(P(\mathcal{U})) = \sum_{i=1}^{n} \log(p(u_i)) \tag{2.4}$$

This is preferred to Equation 2.3 as using addition instead of multiplication results in performance improvements. To convert a `logprob` back into the probability of a token or sequence of tokens, we can simply take $P(\mathcal{U}) = e^{L(\mathcal{U})}$.

The OpenAI parameter for `logprobs` (max 5) gives us the log probabilities for those many most likely tokens as well as the sampled token. By retrieving the `logprobs` for each token, we may better understand the composition and nuances in how the response was built.

In subsequent sections, we experiment with these parameters to better understand the impact on completions.

2.2 Models Simulating Humans

LLMs, by creation, are considered to capture much of the syntax and semantics of human language, as well as a broader representation of human behavior and the world. Depending on the model and if it was trained or fine-tuned for a specific task, LLMs may be used to simulate humans in AI-based experimentation. This idea has been subject to some controversy, and in this section we explore some of the current research and findings in how models are able to simulate humans, as well as why this is significant. We consider the multitude of ethical discussions on the implications of using the results of these simulations in Section 4.2.

Mayo-Wilson [16] suggests that computer simulations should be considered core philosophical methods because they are superior to thought experiments in achieving some philosophical goals. Further, they argue that creating these computational models instill good philosophical habits and that they should be used alongside other tools such as thought experiments and empirical research rather than replacing or supplanting them altogether.

Using AI-based simulations in economics and social experimentation provides a streamlined, cheap, and efficient way to compare multiple experiment methods and gain insight into potential human behavior. By running several inputs on an LLM, researchers can analyze the different results to understand which prompts lead to the least bias or which prompts are the clearest to understand. These can also be run very quickly and are much cheaper than using human subjects as test subjects.

Economists tend to place less interest in simulations using AI-based agents and argue that such experimentation would have less value to economics. One main objection is whether such models actually are able to represent human behavior in a way that allows results to be considered significant [12].

However, several recent experiments using modern LLMs demonstrate that lan-

guage models may be conditioned to respond to inputs with outputs that are consistent with their endowments, including similar biases and perspectives. Here, an endowment refers to a phrase or set of text prompting the AI agent to hold a set of beliefs or traits similar to a human. Using these, it is possible to simulate agents of various genders, races, and political beliefs, and to retrieve responses that reflect those distributions. Argyle et al. [4] defines *algorithmic fidelity* as the measure of how accurately a model replies consistently with the conditioning, and *silicon sampling* to be how LLMs can generate a large sample of AI-based agents. Further, they provide evidence of algorithmic fidelity in GPT-3 within the context of general public opinion as well as how it may be applied to social science research.

2.2.1 AI Agents Replicating Human Behavior

Now that we have explored *why* it would be helpful and possible to use models to simulate humans, we try to understand *if* AI agents are able to replicate human behavior especially as our library aspires to use LLMs as people in running experiments. A wide array of GPT-3-based studies have tested the extent to which LLMs can replicate human behavior in social and economic situations. Argyle et al. [4] determined that a language model, with the correct conditioning, can replicate biases against specified groups in manners that correspond to established human response patterns. Binz and Schulz [5] used vignette-based and task-based experiments to test how GPT-3's behavior compares with human subjects. The tasks tested fell into three categories: decision-making, information search, deliberation, and causal reasoning. They concluded that responses to vignette-based experiments were susceptible to perturbations by slightly altering the input vignettes, while the model was able to solve most of the tasks presented.

Horton [12] tested GPT-3 using classical and modern behavioral economics experiments, concluding that the discrete behavior was similar to documented human responses. Brand et al. [6] studied distributions of responses to study questions about changes in the attributes of goods on choice probabilities and found that the results were consistent with recorded consumer behavior, noting patterns such as

downward-sloping demand curves and state dependence.

Finally, Park et al. [18] replicated 14 experiments from the Many Labs 2 project [14], a large-scale psychology study replication project. They noted how the use of GPT-3 and silicon sampling avoided traditional responses biases, such as non-response bias and exclusion bias. Amongst the 8 experiments that were analyzed, GPT-3 replicated 37.5% of the original results, compared to the 50% replicate rate achieved by the Many Labs 2 project. However, for the other six experiments, the researchers noted a "correct answer" effect, where GPT-3 responded to prompts with near-zero variation even in questions where humans responded with significant variation. Park et al. determined that GPT-3 cannot be assumed to generalize to and replicate human behavior, but the potential for LLMs to automate human economic activity necessitates understanding these models. Although they conclude that studying LLMs should not be used to study the psychologies of humans, we argue there is significant value in having a way to preliminarily test responses to proposed surveys and experiments in order to create early benchmarks, especially given recent research supporting GPT-3's successes in replicating these various studies accurately and realistically. Our aim in this thesis is to create a tool that allows us to more achieve this goal in an easier and more streamlined way, accessible to any researcher interested in using models to replicate human behavior regardless of technical skill or prior knowledge of natural language processing.

2.2.2 Issues and Considerations

Park et al. [18] identified several issues with using language models as a method of simulating human behavior. In this section, we delve into further issues to consider prior to designing the package.

Horton [12] and Binz and Schulz [5], among others, identified a performativity problem, where it can be argued that LLMs and specifically GPT-3 may have encountered several of the tasks and situations used in the replication studies during its training, thus "memorizing" the results. However, GPT-3's responses has been shown to replicate behavior aligning to these theories, rather than completely reciting

them [12].

GPT-3's responses are highly susceptible to subtle changes in prompts, giving rise to the field of prompt engineering [23, 21]. How a prompt is presented may significantly alter GPT-3's outputs. We study variations in prompt structure as we design the package MVP, to provide further understanding of how these differences arise.

Finally, even models as advanced as GPT-3 are still subject to the established shortcomings and inaccuracies — from susceptibility to minor changes in prompts to holding biases from the data using in training. They are not automatic replacements for human samples in experiments, and it is difficult to replicate the same variety of nuances present in people. However, LLMs may be used to test potential prompts and create a benchmark for predicted human behavior.

Chapter 3

Development

Our primary goals in developing our Python-based library, called `homo silicus`, was to allow users to quickly create AI agents and use them in simulated economic experiments and social scenarios. Our first goal was to develop streamlined functions and modules to allow experimenters to efficiently outline different scenarios and AI agent endowments, which are combined and sent to available APIs for responses.

Horton open-sourced the code for the experiments run in his original exploration using LLMs as simulated economic agents [12]. The simulated experiments were Charness and Rabin [8], Kahneman et al. [13], and Samuelson and Zeckhauser [20]. Our initial tests were to replicate the prompts and endowments created for these experiments and to analyze the outputs returned by GPT-3.

Following these trials, we aimed to polish and formalize the package into a minimum viable product (MVP) and seek feedback regarding the preliminary design considerations. Future versions iterated on the base `homo silicus` library in order to create a better user experience. In the following sections, we discuss the original design considerations and explore each package version's performance and results.

Table 3.1: Basic requirements for preliminary `homo silicus` modules

Model	1. Provide options to select specific model 2. Query corresponding API with prompt/s for each experiment iteration and subject 3. Asynchronously retrieve and store results
Experiment	1. Create experiment skeleton 2. Frame experiment with relevant prompts 3. Compile prompts, including subject endowment
Subject	1. Add support for age, gender, race, income, name, and political alignment 2. Generate endowment prompt for given subject

3.1 Design Considerations

3.1.1 Requirements

Our requirements covered three areas: model, experiment, and subject. Individual details are covered in Table 3.1.

Using these modules, we are interested in allowing for users to create a set of prompts that are assembled flexibly so they can be structured as necessary, and then each run through the selected models. Specifically, in the MVP, our goal is to encourage flexibility in order to be able to replicate as wide an array of experiments and subjects as possible. Once the prompts for each are defined by the user, each combination is input to the models using the API, and the results are then processed and saved for analysis.

3.1.2 Current Industry Tools

To understand how best to create the structure for the experiments, we turned to current industry tools for running economic experiments. Current survey technologies include Qualtrics and Amazon Mechanical Turk, which are more simplistic but do not have any inherent support for the desired requirements like being specifically designed for economic experimentation. Tools suited for more economic-specific research include oTree [9], LIONESS [10], and Empirica [3]. These are platforms or

Table 3.2: Structure of oTree experiments [9]

Session	A session is a series of subsessions
Subsession	A subsession contains multiple groups
Group	A group contains multiple players
Player	Each player proceeds through multiple pages

packages designed to run and support economic and social experimentation.

The first package, oTree, is a framework based on Python that lets users build controlled behavioral experiments in economics or psychology, and multiplayer strategy games, like the prisoner's dilemma, public goods game, and auctions [9]. An overview of the hierarchy of oTree entities is shown in Table 3.2.

Empirica is a JavaScript-based framework for developing and deploying multiplayer interactive experiments and games to be run in the browser. It creates a polished interface for running many experiments, with real human participants joining and partaking in the experiment. Figure 3-1 shows the structure of an Empirica game.

We focused on Empirica's components in further designing the structure of an experiment in `homo silicus`. First, Empirica allows for Intro Steps, which include the instructions and any other preliminary prompts. Next, the players enter the main experiment by completing each Stage of each Round. Finally, when the last Round ends, the players complete the Exit Steps.

This structure is very similar to oTree, but includes more flexibility in adding steps for the introduction and conclusion. Both tools provide a great framework for designing experiments, and we take inspiration from both in structuring the `homo silicus` modules. A key difference is that these technologies assume the participants will be selected and entered by the experimenter themselves, and therefore do not provide support for recording or adding treatments for subject endowments unless explicitly added as a form by the researcher. However, our package must support a subject module to develop endowment prompts for simulating unique participants.

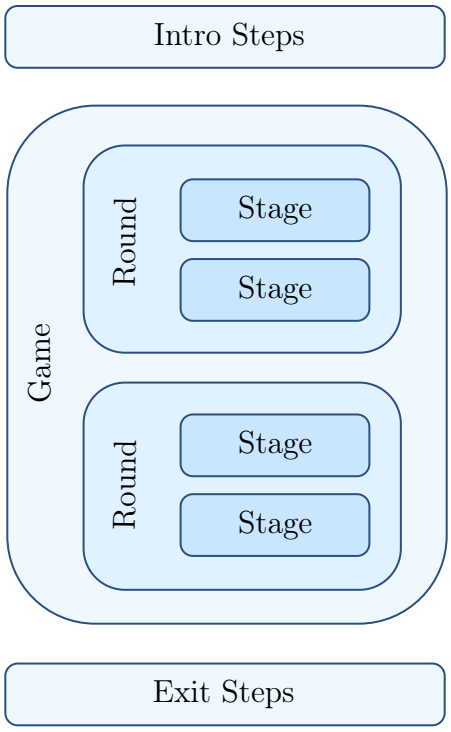

Figure 3-1: Structure of Empirica games [3]

3.1.3 Prompt Engineering

Prompt engineering is a growing area of research, focused on optimizing descriptions of tasks that an AI, such as a large language model, is supposed to accomplish. Understanding the best prompt structure for a given type of economic experiment is crucial for receiving accurate and representative outputs. A key reason for designing the MVP to be as flexible as possible was to allow for testing of various prompt structures. This way, we can understand which prompt structures provided results most similar to established experiments or the most consistent outputs.

White et al. [23] define categories of various prompt patterns, two of which align with our preliminary requirements in Table 3.1. Specifically, the "Output Automater Pattern" may be applied to the experiment actionable itself, and the "Persona Pattern" may be applied to the subject endowment. Additionally, Wei et al. [22] suggested using *chain-of-thought* prompting to improve complex reasoning in LLMs.

We incorporate these patterns as well as variations in the prompts to understand

the influence of prompt structure for each experiment, tested in comparison with original experiments as well as consistency.

3.2 Minimum Viable Product (MVP)

Incorporating our preliminary research and requirements, we created a minimum viable product (MVP) to structure and reduce the amount of code needed for a user to replicate the experiments conducted by Horton [12]. In this section, we outline the design of the MVP and experiments we ran using it.

3.2.1 Modules

Within the `homo silicus` package, there are three main modules: Model, Subject, and Experiment, as shown in Figure 3-2; these mirror the requirements outlined in Table 3.1. These are then used to build streamlined scripts, and using the scripts, we recreated the experiments originally replicated by Horton [12].

The `Model` class initiates preliminary parameters for creating an GPT model and setting necessary error-handling variables. The class method `run_prompt(...prompt...)` accepts prompts as a parameter, which are then sent as input to the selected OpenAI models.

The `Experiment` class sets all rounds, subjects, and endowments, and it combines these iteratively to build all of the possible experiment prompts. Within the `Experiment` module, we also have a `Round` class which holds an isolated set of tasks and choices and generates a corresponding prompt. The choices are given as a list, and in the generated prompt, they are assigned a numerical value. The model is then asked to select on number; this constraint optimizes for a succinct and clear response from the model. Multiple rounds may be added to an experiment, each with their own class `task`, `transition`, `instructions`, and `choices` variables. The `Round` class has a class method to `generate_round_prompt()`, which is used to create a corresponding prompt for that round.

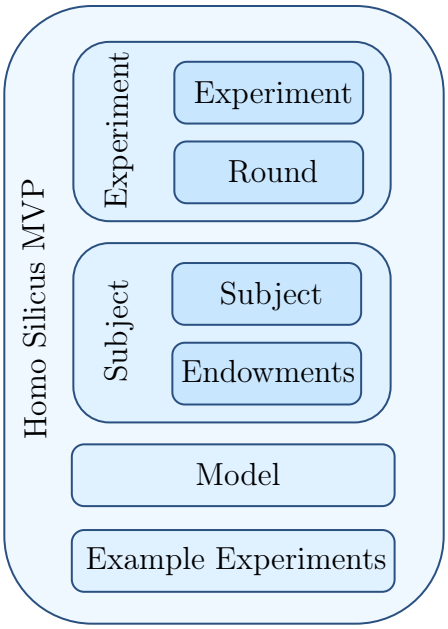

Figure 3-2: Structure of `homo silicus` MVP package

The `Subject` class creates a mock AI subject and contains functionality to generate a corresponding prompt with the given attributes. These attributes are set using the `Endowments` class, which holds all the available parameters used to set an AI subject's "personality". For the MVP, there is support to integrate attributes for name, age, gender, race, income, and political alignment attributes. Additional attributes must be provided as lists of full sentences that are appended to the endowment prompt.

In generating the prompt, we combine each of these classes' respective prompt generator, which are then used as input for the LLMs. There is minimal post-processing of the results in the MVP, and they are output for the researcher to use. A baseline prompt structure is shown in Table 3.3. In testing, we chose to focus on the first three experiments replicated by Horton in the original paper, as this type of work is what we aim to streamline in the package. Replication experiments are run at `temperature = 0`, unless otherwise stated.

Table 3.3: Structure of `homo silicus` MVP prompt

Experiment - individual prompt
Introduction
Subject Endowment Prompt
Preset parameters (age, gender, etc)
Additional parameters
Round 1
Task
Transition
Instructions
Choices
"What choice... [pick one word/number]"
Round 2...

3.2.2 Replicating Charness and Rabin

The first study we reproduce was Horton's replication of Charness and Rabin's experiments on the two-person dictator game, where a subject has to choose between two allocations for themselves and another participant to test their choices regarding efficiency and equity [12, 8]. We use the same labels for each set of tasks, which were set in [8] based on the location of the original studies and specific treatment.

Below, we show a comparison of the original prompt created for the Berk29 treatment, followed by a prompt created by `homo silicus`. The code itself was around 50 lines, compared to the original file from [12] at around 130 lines. Both experiments were run with `temperature = 0`, outputting results as deterministic as possible.

```
You are deciding on allocation for yourself and another person, Person A.
You only care about your own pay-off


    Option Left:  You get $400, Person A gets $400
    Option Right: You get $375, Person A gets $750


    What do you choose, with one word [Left, Right]?
```

```
You are a person named Subject 3. You only care about your own pay-off.
You are deciding on allocation for yourself and another person, Person A.

0) You get $400, Person A gets $400
1) You get $375, Person A gets $750

What is your choice, with one word: [0, 1]:
```

There are some differences between the prompts due to encouraging flexibility in future use cases. For example, we add a line declaring to the model, "You are a person named Subject 0." This is in line with the "Persona Pattern" outlined by White et al. [23] and is expanded upon when an automatically-formatted parameter is given (age, gender, race, income, political alignment, name). In this case, the endowments are entered as an additional parameter, and appended to the subject-level prompt.

These prompts are generated from a list of treatments, where the amounts of money available for the subject to choose between vary. The available choices are automatically given a numerical index to aid the model in giving a clear, succinct answer in order to aid in building an automated post-processing of the results.

The results from testing these prompts are displayed in Figure 3-3. For the results found by `homo silicus`, several key post-processing steps are manually done in the MVP to clean and process the data - in future iterations, we iterate on these steps and integrate them further into the package itself. First, we retrieve the text output and identify if it is a number ("1") or a word ("One!"). If the output is a word, we strip non-alphanumeric characters and try to convert the word into a number. Using the original result, we capture the corresponding log probabilities and convert it back into a probability, which we used as a proxy measure of confidence when displaying results in Figure 3-3. We explore how `logprobs` compare when varied with temperature and number of runs in Sections 3.2.5 - 3.2.7.

Comparing the results, we find similarities across both versions of the replications. For the first three endowments varying the subject's priorities, the outputs are exactly

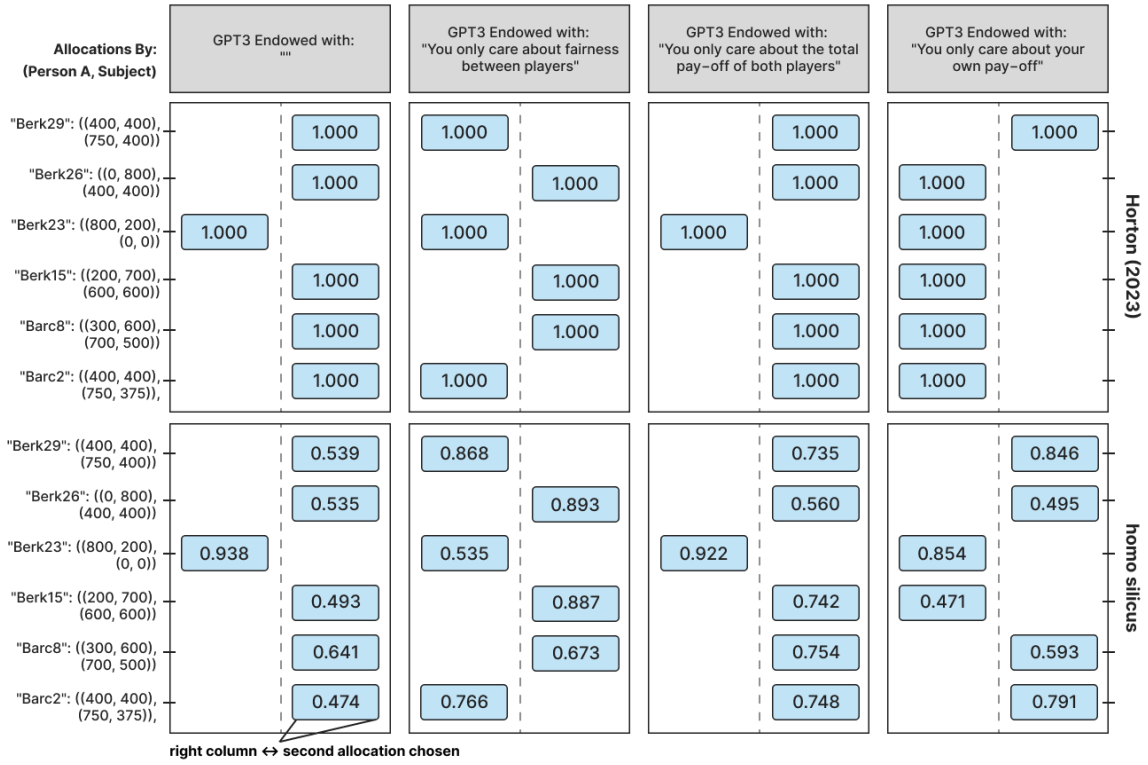

Figure 3-3: Charness and Rabin replication tests completed by `text-davinci-003` varying treatment and endowed "personality", run by Horton [12] and `homo silicus`

the same, with the `homo silicus` package recording mostly > 0.5 probabilities. For the "selfish" endowment, the `homo silicus` results only line up with 3 out of 6 of the results recorded by Horton. Significantly, the results are less inline with the given endowment comparatively, which may indicate a problem with the prompt construction. In subsequent experiments, we test the effects of alternative prompt structures and randomized the choice orders.

3.2.3 Replicating Kahneman et al.

Kahneman et al. [13] studied intuitions about market fairness in response to market scenarios regarding a hardware store raising snow shovel prices after a snowstorm. Horton expanded on this study by adding dimensions to the treatments for the price the shovels were raised to as well as framing of the prompt ("changes the price to" vs "raises the price to"). We replicated this study using `homo silicus` with the recreated prompts. Using the MVP-level prompt construction, the results are displayed in Figure 3-4. Horton's results, when run again to build the given graphics, were exactly consistent with those presented in the paper [12].

In comparison to the Charness and Rabin study, the results using `homo silicus` differed significantly from Horton's replication; 30 out of 48 of the results were different, with 4 of them different by 2 of magnitude (ex. "Acceptable" vs "Very Unfair", where "Completely Fair" was never chosen) Critically, some of the trend in the responses seemed improbable — for example, the leftist AI subject found the price increase from \$15 to \$20, framed as a price "raise," to be "Very Unfair." However, for the same framing but with a higher price increase from \$15 to \$40, the leftist subject found it to only be "Unfair." This is not in line with the trends noted in either of the previous studies, leading us to iterate on the prompt creation process in order to better understand some of the factors influencing the results.

Specific improvements to the prompt included only optionally setting a name for the AI subject (previously, was forced to be endowed with "Subject 3" etc.) and changing the prompt numbering to be 1-indexed rather than 0-indexed. Examples of each type of prompt are provided in the Appendix A.1.1. Results using the improved

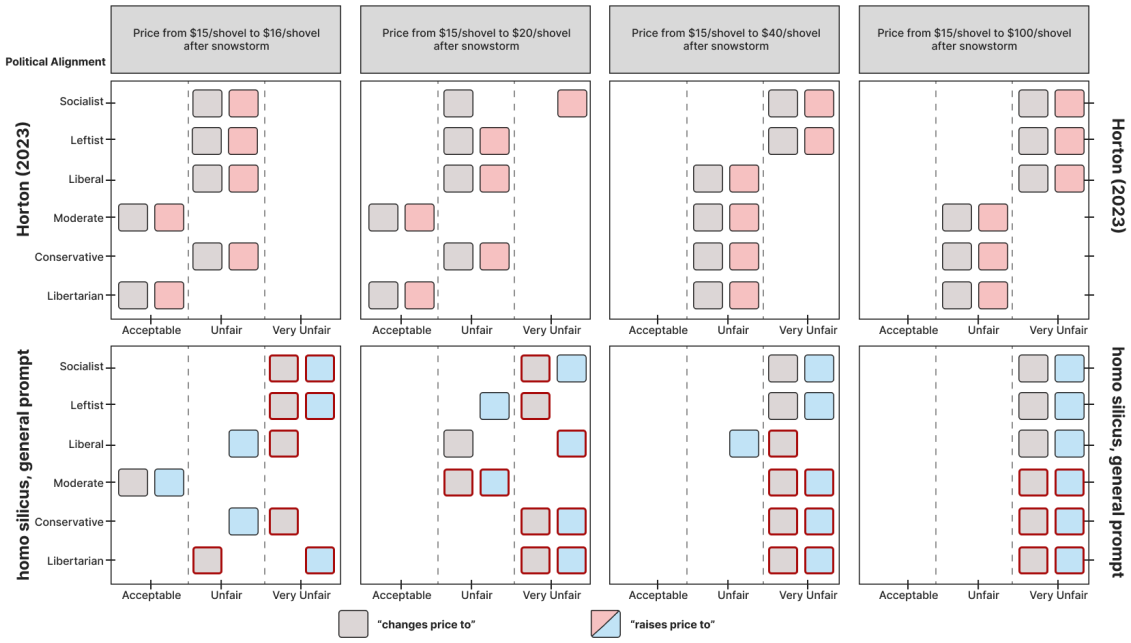

Figure 3-4: Kahneman et al. replication tests completed by `text-davinci-003` varying treatment and endowed political alignment, run by Horton [12] and `homo silicus`. Red-outlined cells are choices misaligned with Horton's replications.

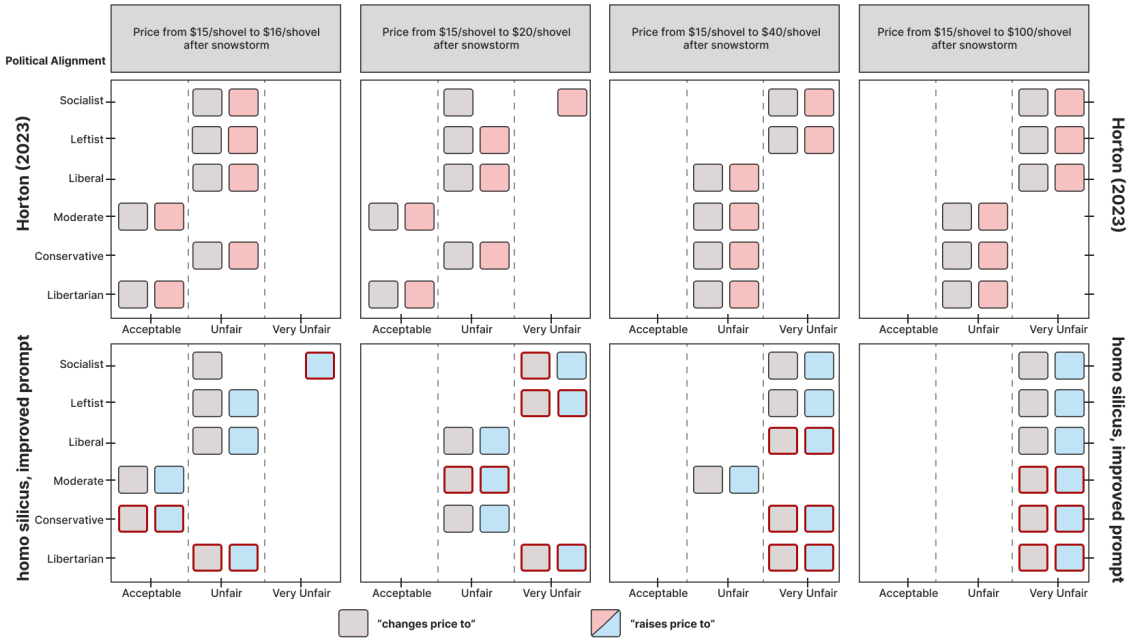

Figure 3-5: Kahneman et al. replication tests completed by `text-davinci-003` with a manually improved prompt structure. Red-outlined cells are choices misaligned with Horton's replications.

prompt construction are displayed in Figure 3-5.

This time, 24/48 of the results were different, with 2/48 different by 2 or more orders of magnitude. The trends seem more consistent, with the sense of unfairness increasing with the price across treatments. However, the high proportion of differing results compared to both of the previous replications demonstrates the significant impact that prompt construction and ordering has on the LLM's responses (especially since the content was nearly word-for-word identical).

3.2.4 Replicating Samuelson and Zeckhauser

The last external study replicated by Horton was Samuelson and Zeckhauser's [20] work on understanding status quo bias, focusing how an option is more likely to be selected if it is presented as the status quo. In this study, subject were asked to choose allocations for car and highway safety budgets. The AI subjects were also endowed with specific preferences towards either car or highway safety, or neither. The prompts for both the original replication and the `homo silicus` experiments are given in Appendix A.1.2. Each prompt contained introductory information regarding the budget itself, the current budget allocation, budget choices, the subject's view/endowment, and the task on choosing on of the choices. The choices themselves were either to maintain the present budget amounts, or to adjust the current budget allocation to one of the following: (70% cars, 30% highways), (40, 60), (30,70), and (50, 50).

Each experiment was run using one AI subject, setting `temperature = 0` to ensure the most deterministic result. The results are given in Figure 3-6 (red-outlined cells are allotments most in-line with the given endowment). After several initial runs, we observed that the results between the two replications differed much more than in previous experiments. Therefore, for the `homo silicus` experiments, the phrase "[choose] with one word" (intended to prompt the LLM into only returning one-word/character, easily parse-able results) was removed. This change resulted in GPT-3 adding more text than just the choice chosen in 18/25 of the responses (ex. "4" vs "4. I believe that car safety and highway safety are equally important and should be

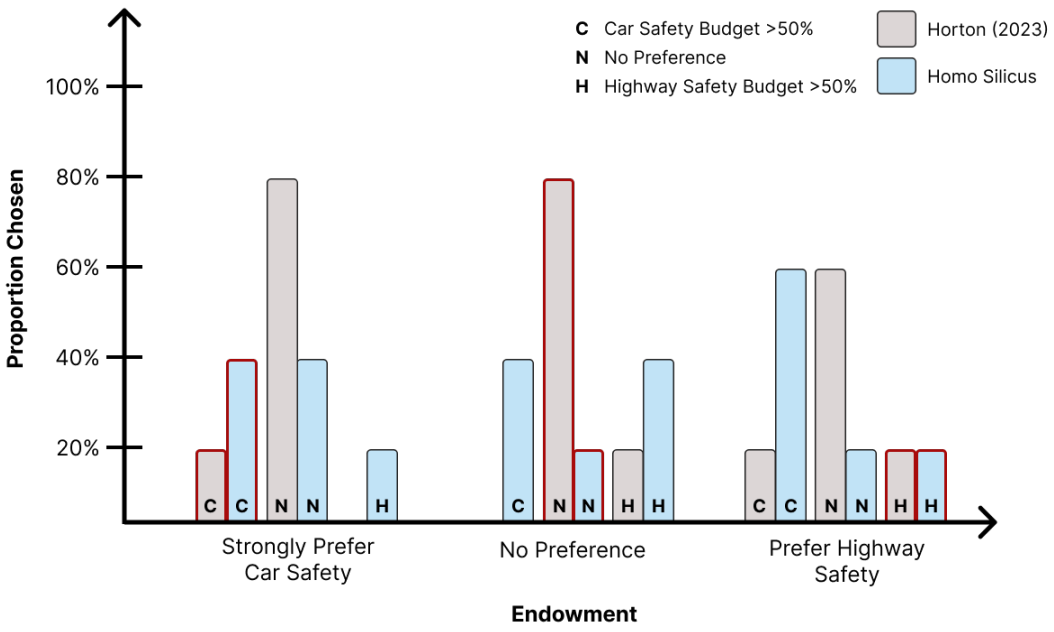

Figure 3-6: Samuelson and Zeckhauser replication tests.

allocated the same amount of resources"). Often, this additional text rationalized or explained the choice, which we analyzed to better understand why the results between the two replications were so different.

Horton observed that the 50/50 split was most common in the neutral framing, and in the cases where one option was presented as the status quo, it was more commonly chosen. We noticed similar trends — when the model was endowed with no preference between car and highway safety, each of the responses made the choice to maintain present budget amounts/the status quo.

There were some inconsistencies in the reasoning given in the `homo silicus` experiments. For example, in an experiment with the current allotment at 70% to auto safety and 30% to highway safety and the view "highway safety and car safety are equally important," the response chose to decrease auto program by 10% of budget and raise the highway program by the same amount. In the rationale, the AI subject reasoned:

> Highway safety and car safety are both important and should be given equal consideration. By decreasing the auto program by 10% of the

> budget and raising the highway program by the same amount, the National Highway Safety Commission can ensure that both programs are adequately funded and that both safety concerns are addressed.

However, although the chosen option does bring the split from 70/30 to 60/40, there was an option to decrease auto program by 20% of budget and raise the highway program by like amount which would have lead to a 50/50 budgetary split to truly ensure both highway and car safety would "be given equal consideration."

Additionally, the results in [12] were slightly more inline with the given endowment than in the `homo silicus` experiments. In comparison, the results we observed were more distributed regardless of the stated view. In one case, the reasoning provided by the AI subject was opposite to the endowment given in the prompt. The view was to prefer highway safety ("highway safety is slightly more important than car safety") with a current budget allocation at 50% to auto safety and 50% to highway safety. The result chosen was to decrease the highway program by 10% of budget and raise the auto program by the same amount, with the pro-car safety reasoning:

> Highway safety is important, but car safety is also important and should be given more attention. By decreasing the highway program by 10% and raising the auto program by the same amount, both programs will still receive adequate funding while giving more attention to car safety.

This inattention to the endowment with the `homo silicus` experiments may possibly be because the endowment was at the beginning of the prompt, rather than towards the end. In turn, GPT's self-attention mechanism may give less weight to the endowment due to this earlier position. Specifically, GPT-3 uses this mechanism to give varying weight to different parts of the input data. By doing this, the model weights different parts of the sequence in order to infer meaning and context. By default, earlier tokens are not given less weight than later tokens — however, the model may assign higher weights to tokens that are closer in position to the current token being processed [7]. In order to better understand how prompt order influences results, we tested the effects of alternative prompt structures in the following Section.

Table 3.4: Various prompt orderings tested

Prompt Order 1	1. Subject Endowment 2. Task 3. Choices 4. Question
Prompt Order 2	1. Task 2. Choices 3. Subject Endowment 4. Question
Prompt Order 3	1. Task 2. Subject Endowment 3. Choices 4. Question

3.2.5 Prompt Order

Next, we varied the ordering of the various prompt components in order to compare the impact on results. We focused on the Charness and Rabin experiments, as our replications for these were most consistent with Horton's established results. The prompts for these experiments may be broken up into sections consisting of: (subject endowment, task, choices, question). This order mirrors the first prompt we tried, whose results are discussed in 3.2.2. We ran the experiments again, varying this in two other reasonable orderings for the prompt: (task, choices, subject endowment, question), and (task, subject endowment, choices, question). These are listed in Table 3.4 and in Appendix A.1.3.

The results are given in Figure 3-7, including the choice probabilities and in comparison to the results in [12]. Each experiment was run once (i.e. with one AI subject) but with `temperature = 0`, so the results were as deterministic as possible and re-running provided the same choices with very similar probabilities. They could not be made completely deterministic through the API.

Prompt order 3 (in Appendix A.1.3) was most similar to Horton's prompt ordering (task, subject endowment, choices, question). Both orders 2 and 3 have more varied inconsistencies, whereas the original prompt order 1 was consistent with Horton's

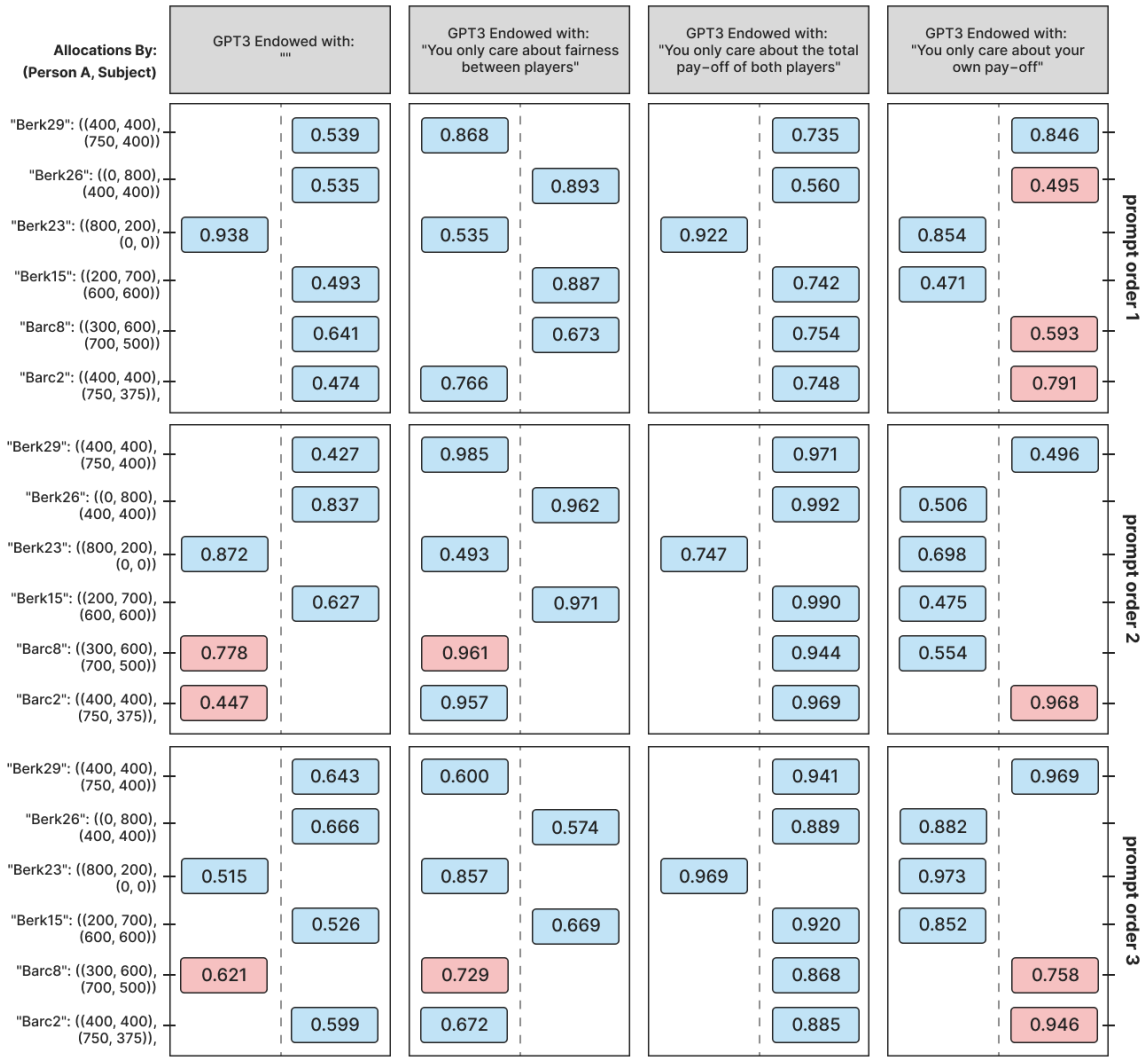

Figure 3-7: Charness and Rabin replication tests completed by `text-davinci-003` varying prompt orderings. Red cells are choices misaligned with Horton's replications. Prompt orders are given in Appendix A.1.3.

results in every endowment except in the selfish one. Because order 1 was more consistent, flowed the most intuitively and was most in line with the Persona Pattern (give endowment, and then task), and had the least deviations from Horton's results, we chose to maintain this order for the MVP.

3.2.6 Choice Randomization

Prior work identified that GPT-3's responses are susceptible to changes in the prompts, including varying choice orders [22, 25]. We added functionality to randomize the order of the given choices in the prompt, as well as recording the orders to keep track of results. Finally, we compared the results with randomized choice orders to the original `homo silicus` experiments for the Charness and Rabin tests. The results are given in Figure 3-8. We ran each experiment 100 times, and the proportion of times the majority answer was given as well as the `logprobs` are displayed in the data as (proportion, average probability).

Between the 24 unique treatments across the two sets of experiments, there were only 3/24 different responses, in: (no endowment Barc2, total pay-off endowment Berk15, and total pay-off endowment Barc2). As a result, we chose to keep the functionality for randomizing choices in the package, but set the default to not use it.

The averaged probabilities were highly similar across both experiments when the majority choice was the same. The majority of choices were deterministic, with all 100 runs resulting in the same outcome even when this outcome differed between experiments. For example, the Barc2 experiments with no endowment and "total pay-off" endowment resulted in opposite results between the original and randomized sets of tests, with the entire proportion of results choosing the respective outcome. The Berk15 experiment with randomized choice ordering and endowment for "total pay-off" resulted in an exact 50/50 split between the two choices, which was especially interesting as these were run with `temperature = 0` — we would ordinarily consider this very deterministic and expect to see the responses at least lean towards one choice.

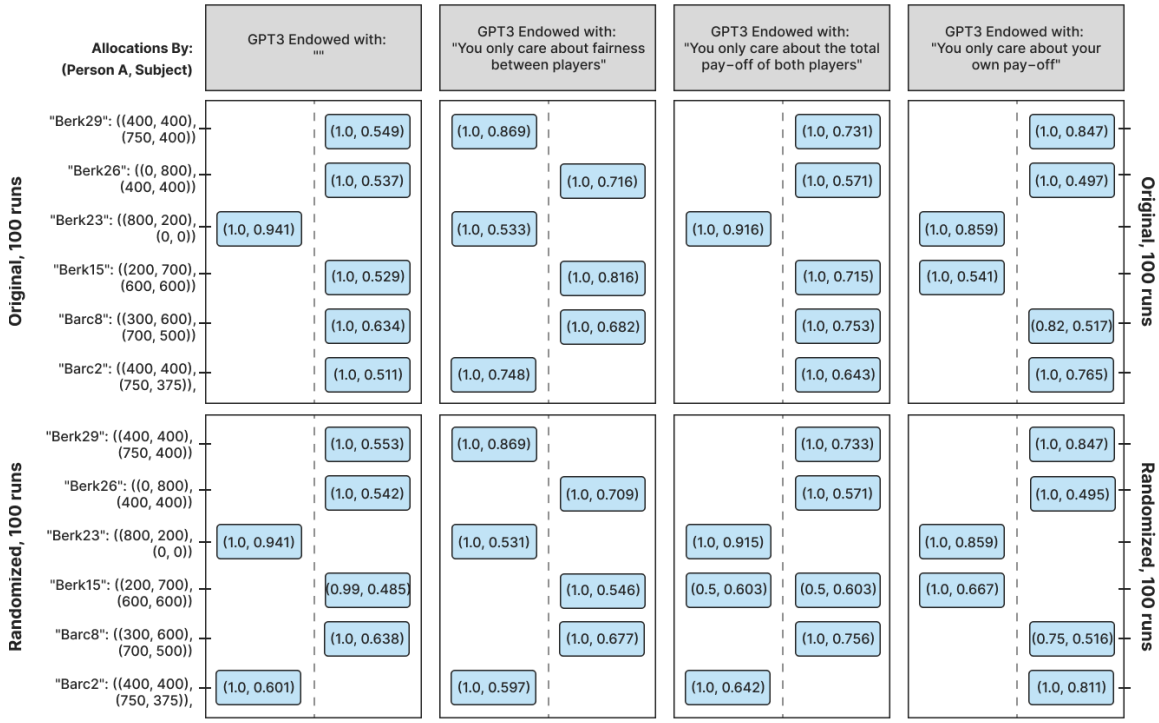

Figure 3-8: Charness and Rabin replication tests completed by `text-davinci-003` testing randomized choice orders, run 100 times.

3.2.7 Interpreting Outputs

In a traditional experiment with multiple human subjects, responses will vary by person. Our previous experiments were run at `temperature = 0`, resulting in responses with little variance, akin to every response being identical. To add more "creativity" in the responses, we may increase the temperature - the default value is 1, with many resources claiming 0.8 results in the optimal variance [18].

We were interested in how this resulting variance may be used as a measure of population proportion choosing a specific option, especially compared to the token probabilities we previously used as measures of confidence. In order to better understand the relationship between the token `logprobs` and the choice answer frequency, we ran experiments comparing these across various temperatures. Because the Charness and Rabin experiments with the "total pay-off" endowment was most stable across prompt orderings and identical to Horton's replication tests, but varied most with randomized choices ordering, we chose to focus on this treatment first and later tested the same relationships with endowments "total pay-off" and "your own pay-off."

Through the OpenAI API, we are able to access and retrieve the probabilities for each token in the final response. Assuming that the corresponding number for the AI subject's choice is present in the response and given the trend identified in previous experiments, we assume that the first non-whitespace token is either the numerical or word form of that number. Using this, we may retrieve its log probabilities and calculate the probabilities as shown in Equation 2.4. This calculation is often used as a measure of confidence, and we were interested in whether this captures population proportion. Additionally, we found the frequency at which the top choice was selected across the runs which would be more similar to traditional measures of population proportion, treating each run with an AI subject as an individual person.

For each treatment, we ran the experiments at temperatures [0, 0.5, 0.8, 1.0, 1.5, 2.0], and with 100 runs. We then averaged the probability (`avg_prob`) and calculated the answer frequency (`ans_freq`) for the top choice (the top response for

Table 3.5: Example of output data. (`ans_freq`, `avg_prob`) pairs for `temperature = 2` across all allotments and chosen endowments from the Charness and Rabin replications. Similarity scores are calculated using Equation 3.1
.

Allotment/Endowment	""	"total pay-off"	"your own payoff"
"Berk29": ((400, 400), (750, 400))	(1.0, 0.571)	(0.98, 0.627)	(0.979, 0.514)
"Berk26": ((0, 800), (400, 400))	(0.986, 0.447)	(1.0, 0.409)	(0.814, 0.278)
"Berk23": ((800, 200), (0, 0))	(0.958, 0.672)	(0.969, 0.686)	(0.948, 0.540)
"Berk15": ((200, 700), (600, 600))	(1.0, 0.467)	(1.0, 0.449)	(0.571, 0.190)
"Barc8": ((300, 600), (700, 500))	(0.988, 0.418)	(0.866, 0.314)	(0.648, 0.242)
"Barc2": ((400, 400), (750, 375))	(1.0, 0.539)	(0.937, 0.568)	(0.944, 0.390)

each treatment were the same as those recorded in Figure 3-3). To get the similarity between these two scores, we calculated

$$\texttt{similarity} = 1 - |\texttt{avg\_prob} - \texttt{ans\_freq}|. \tag{3.1}$$

The results are displayed in the top plot of Figure 3-9. The calculation for comparing these two measures of response proportion did not take into account whether the answer itself for each were different, but separately analyzing the responses showed that across all temperatures, the majority choice for final responses were the same and overall identical to those in Figure 3-3. Given that the instructions specifically tried to constrain the responses to give the choice using the phrase "with one word" in the instructions, it was interesting to see GPT try to get around this at higher temperatures by outputting responses such as "giventheallocationconditionshereone-choice1" and "perobjectorsumptionutive2." We did not use the responses for runs where this phenomena occurred, so at a temperature of 1.5 about 0.5% of responses were not considered in calculating final proportions; at a temperature of 2, about 3% of responses were not used.

There was a general trend of a higher temperature leading to a lower similarity between the two proportion metrics. We noticed that even at `temperature = 2`, the response frequency tended to be very high (all >0.9). We specifically ran these

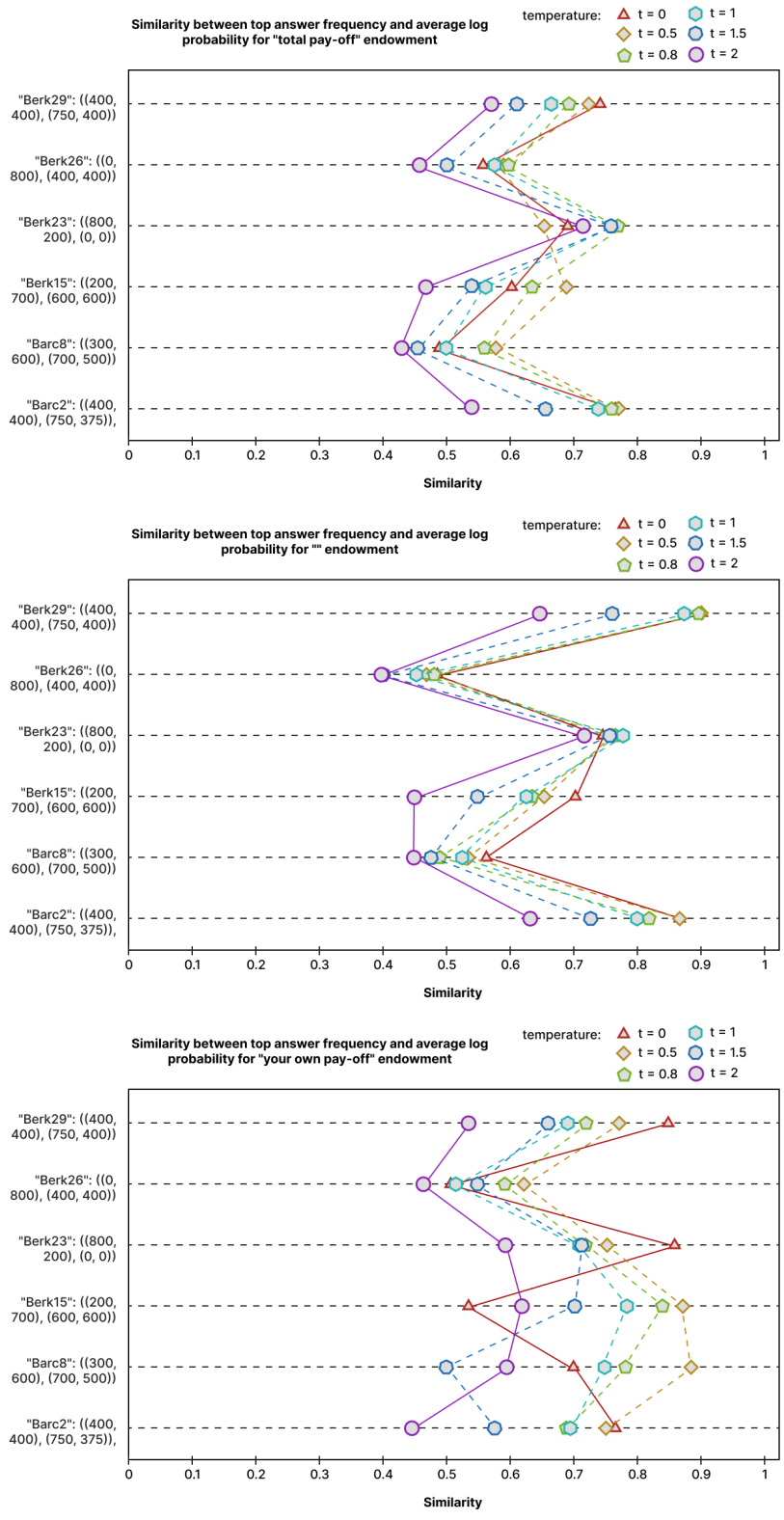

Figure 3-9: Replication test at various temperatures for Charness and Rabin experiment with different endowments, comparing similarity between top answer frequency to average log probability for each scenario.

experiments with the "total pay-off" endowment because it was the most stabled across previous experiments, but we considered this may have lead to a higher degree of confidence than most other prompts. As such, we ran the experiment again, this time varying the prompt to have no endowment ("") which yielded the same responses as observed by Horton [12] but was less stable across prompt orders, and the "your own pay-off" endowment, which is the least stable endowment.

The second and third plots in Figure 3-9 displays the results for these two endowments. We see similar trends but with greater variance of similarity scores within allotments. For example, in the "total pay-off" endowment testing the Berk29 allotment, the maximum difference between similarities was 0.170, between `temperature = 0` and `temperature = 2`. For the same allotment using the ""/empty endowment and "your own pay-off", the maximum difference was between the same temperatures and was 0.261 and 0.315, respectively. Significantly, the orders of plots within each allotment "line" was similar across experiments — again for the Berk29 allotment, similarity strictly decreased as temperature increased across all endowments.

A general trend we noticed was that lower temperatures had a higher similarity score, with the `temperature = 2` score usually displaying very low similarities. The `temperature = 0.5` parameter yielded the highest similarity scores in general. In the future, it would be interesting to factor in the response itself in better understanding whether answer frequency or log probability is a better estimate of confidence and/or population proportion choosing that response.

3.3 Version 2 (V2)

After building and experimenting with the MVP of `homo silicus`, we solicited feedback and potential improvements through informal and qualitative user tests.

We incorporated this feedback in building a Version 2 of the library.

3.3.1 Feedback and Improvements

Users primarily commented on the flexibility of the package — with some concerns about over-flexibility. Although we designed the MVP specifically to give a high degree of control to researchers in how prompts were built and experiments were run, feedback indicated this may create a learning curve in understanding how to use the library effectively. Rather, it was recommended to add more constrained classes incorporating the current modules so that the only code necessary from the researcher is the text for building prompts. Additionally, by creating specialized classes, we can represent more unique types of economic experiments and reduce the workload to build and run them.

Another recommendation was to extend the post-processing pipeline to incorporate data storage and analysis. In the MVP, results were output as dictionaries that would have to be parsed by the researcher to analyze themselves. Users mentioned that this somewhat negated the benefits of using `homo silicus`, as the experiment results would still require iteration and modification.

Allowing experiment parameters to be easily shared between researchers to be immediately run was another suggestion. If these parameters can be stored in a common file type, researchers may easily send the file to collaborators, who can immediately run the same experiment or manually edit the parameters. The JSON file type was specifically cited as a flexible and common method for storing parameters.

Finally, there were some miscellaneous functions and options we developed while building the replications and experiments. We considered which we were interested in maintaining and further incorporating.

Overall, the main improvements we considered from the feedback were:

1. **Specialized classes**: For building specific types of experiments

2. **Post-processing**: Automatically storing and analyzing results

3. **JSONs**: Converting experiment parameters to and from JSONs

4. **Miscellaneous functionality**: Randomized choice ordering, "one word", GPT-4, allowing multithreading, multiple runs, varying temperatures

We discuss how we iterated on these recommendations below.

Specialized Classes We developed classes representing specific types of experiments. This would tighten the flexibility available to experimenters when using these classes, but allow us to conduct more specialized experiment design and analysis of the results. Ultimately, our automation is constrained since all of the experiment-specific information must be written and formatted by the researcher. However, our aim was to streamline the process of piecing these modules together in a simple and understandable manner, and to handle as much of the following pipeline once the prompts are assembled.

The first class we decided to incorporate was a `Surveys` class — each of the three main experiments we replicated for the MVP (Section 3.2.6 Charness and Rabin, Section 3.2.3 Kahneman et al., Section 3.2.4 Samuelson and Zeckhauser) could be represented and recreated as a `Surveys` experiment. Code samples comparing the process of writing these experiments from the MVP to V2 are available in Appendix A.2.1.

The next class we were recommended to develop was a `Conjoints` class. The conjoint analysis, first introduced by Luce and Tukey in 1964, is an analysis technique that measures the preferences of various attributes of a product or service by presenting respondents with different choices incorporating some combination of these attributes [15]. By analyzing their choices, conjoint analysis provides insights into how consumers value trade-offs between different attributes. Green and Wind outlined the application of the conjoint analysis in market research [24], demonstrating how it can be applied to evaluate consumers' judgement and giving examples of its use in comparing flights, toothbrushes, and car tires; in a later article, Green and Srinivasan explored the ongoing development of conjoint analysis in consumer research and explained the *part-worth function* model to quantitatively calculate preferences [11]. For traditional conjoint surveys, the number of potential combinations of attributes

scales exponentially as more attributes and variables are added to the experiment design. Thus, researchers often choose a selection of top combinations to present in human surveys. However, using AI agents, every combination may be tested easily and efficiently. Although it was already possible to run a conjoint survey in the MVP, we developed a specific `Conjoints` class to streamline the process for the researcher.

Our final experimental class was a `Multiagent` class. This was designed for use in iterated public goods games (PGGs), but with the flexibility to be used in other settings as well.

Post-Processing We developed a pipeline for writing results into SQLite databases as the prompts are running, ensuring data storage for future use. Several optional parameters allow users to choose the filename of the database and to select specific databases to create. The general database for every experiment holds the choice and complete metadata for each prompt ran in the experiment, and if the user selects additional databases to create like `endowments`, this adds a table with an additional column storing the corresponding `endowment`.

As we create more restrictive, specialized experimental classes, we have a greater understanding of the types of outputs for each type of experiment. This allows us to add analysis tools directly into the experimental pipeline. For example, if a researcher built a `Surveys` object with the "one word" option (described further below), we could assume the expected output was a number representing a choice in the survey. Using this, we can construct bar graphs and other plots using the outputs from the prompts. Further, if the user chose to create additional tables, these can be used to develop analyses specific to the parameters chosen. Examples of these are displayed in Section 3.3.3.

JSONs An important recommendation was to add the option of creating experiments from JSONs. This allows researchers to simply share a JSON file to a collaborator, who can then replicate the exact same experiment immediately. In order to build an experiment from a JSON, we also added functionality to convert it to JSON

by storing each of the class parameters. An example of the Charness and Rabin experiments ran from a JSON file is shown in Appendix A.2.1.

Miscellaneous Functionality As we tested and experimented with the MVP, we wrote in several additional functionalities to `homo silicus`. We gathered feedback to decide which to carry over the V2.

- **Randomized Choice Ordering**: This allowed users to have the order of the choices randomized when presented to a user. Because LLMs are susceptible to varying choice orders [22, 25] and display a preference for text seen more recently/towards the end of the prompt (as we tested in Section 3.2.5), this may aid in eliminating potential bias. We decided to **keep** this option.

- **"One Word"**: This parameter adds the phrase "with one word" to the end of the prompt, intending to guide the LLM into only returning one-word, easily parse-able results. Setting this to false would allow the model to produce longer rationales for responses — however, we cannot automatically parse such responses when attempting data analysis. We decided to **keep** this option, although removing the phrase also removes options for automatic data analysis.

- **More Models**: As this project developed, access to GPT-3.5 and GPT-4 were released by OpenAI. The API request format varied slightly, and we decided to **add** support for sending prompts to these models into later versions.

- **Multithreading**: We incorporated multithreading in the MVP to speed up testing, especially when completing multiple runs of large experiments. Although this significantly reduced the amount of time required for each experiment, we experienced rate limiting errors from OpenAI. Additionally, we occasionally accidentally ran very large experiments too quickly to realize an error was present, using up money and token allotments. To protect users from these errors, we **removed** the option to use multithreading.

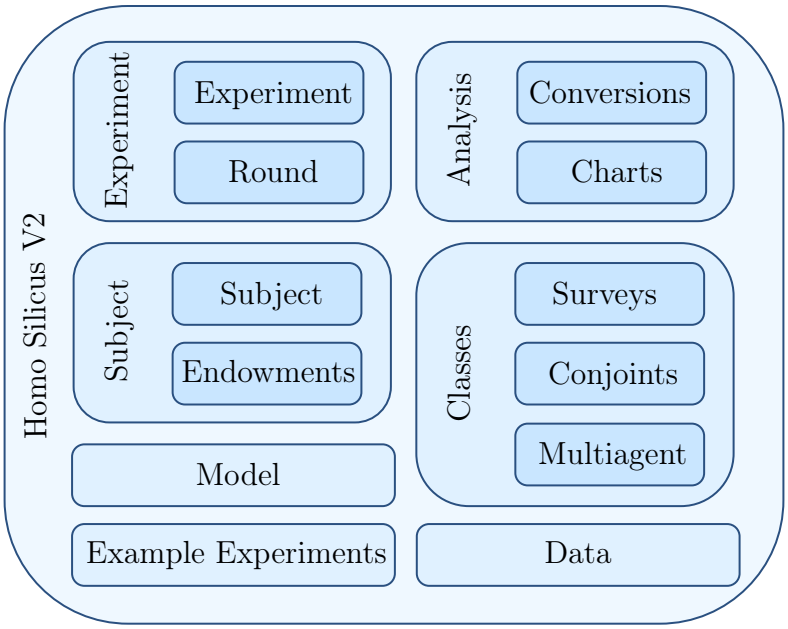

Figure 3-10: Structure of `homo silicus` V2 package

- **Multiple Runs**: We decided to **keep** the option of running experiments multiple times through the same function, as this allows researchers to get averages of experiment results much more easily.

- **Varying Temperatures**: We decided to **add** the option to run experiments at various temperatures, adding it as a parameter to in the specialized experiment class modules and the `Model` module as further described below.

3.3.2 Modules

The fundamental module structure described in Section 3.2.1 was carried into the V2, and we added new modules and parameters building on these. A diagram of the improved, V2 `homo silicus` package is shown in Figure 3-10.

The `Experiment`, `Subject`, and `Model` modules remain on the left side of this figure, and mirror the structure in Figure 3-2. To the original modules, we added support for writing results to SQLite databases, reading from these databases and building visualizations of the results, and defining functions to convert experiments to and build from JSON files. These JSON files are automatically built in `Example`

`Experiments`.

On the right column in Figure 3-2, we see the newly added modules - (1) `Analysis`, (2) `Classes`, and (3) `Data`.

Analysis In the `Analysis` module, we built helper functions for processing and analyzing prompt outputs. The `Conversions` class aids in stripping, cleaning, and extracting specific characters or phrases whenever necessary. For example, if the LLM chose an option in a survey with the response " one! ", the `Conversions` class holds methods that can strip non-alphabetic and non-numeric characters (converts to "one"), as well as take a word or phrase representing a number to the number itself (converts to "1"). Additionally, the `Analysis` module has various classes to build charts and data visualizations automatically. The `Bar Charts` class generates bar charts given data guaranteed to be in a form familiar and parse-able by it. Examples of automatically created charts are displayed in Section 3.3.3.

Classes The `Classes` module holds the specialized classes described in 3.3.1. We will focus on describing the `Surveys` class as it is most similar to the experiments replicated before, but we also support a `Conjoints` class and a `Multiagent` class. In future work, we are interested in further expanding the classes available.

The `Surveys` class limits the parameters that may be used to build prompts. In a general experiment, researchers can add a multitude of parameters that are combined in the order described in Table 3.3. In building a Surveys object, the experimental design is simplified to the `endowment`, `scenario`, `task`, and `choices`; `task` and `choices` are combined to build a `Round`, and `endowment` is used to build a `Subject`. Thus, the three main elements of a survey are `["subject", "scenario", "round"]`. Additional parameters available to use in building and running prompts are `models` and `temperature`.

Each element provided in parameters (`["model", "endowment", "scenario", "task", "choices", "temperature"]`) are combined in using the external `itertools.product` function. Researchers may also customize how the prompt

is built by changing the order of `["subject", "scenario", "round"]`, which is an added functionality not available in a general experiment. The class attempts to clean and process results based on whether parameters such as `randomize_choice_ordering`, `one_word`, and `logprobs` are set to True.

We added functionality to store results automatically using SQLite into a database and create visualizations based on the results. By having specialized classes, we may better understand the format of the expected results and therefore create a more customized post-processing workflow than general experiments. Additionally, we added support for JSON serialization specific to each specialized experiment classes.

The general modules maintain the ability to build the original, general style of experiments as described in Section 3.2.

Data The `Data` folder holds databases and plots built when running the experiments. By default, the filenames used are based on the time the experiments are run to ensure unique names. However, this may be changed using a `file_name` parameter.

3.3.3 Replicating Charness and Rabin as Survey

We replicated the Charness and Rabin experiment once more (originally described in Section 3.2.2), using V2's new `Surveys` class. Breaking the original prompts into `["endowment", "scenario", "task", "choices"]` modules, we could feed all of these automatically into a new `Surveys` object along with parameters for `one_word=True, temperatures=[0, 0.5, 1], logprobs=3`. We tested writing this object into a JSON file, then reading and rebuilding the experiment from the file, running the built prompts, and automatically analyzing the results. Code samples for both of these processes are displayed in Appendix A.2.1.

Examples of charts auto-generated by the class are displayed in Figure 3-11. For every additional table the researcher chooses to create, the corresponding results may be analyzed and visualized. In this experiment, additional tables and graphs were built for the endowment, temperature, and choices parameters. By producing these charts, users may immediately glean possible trends before running their own analysis

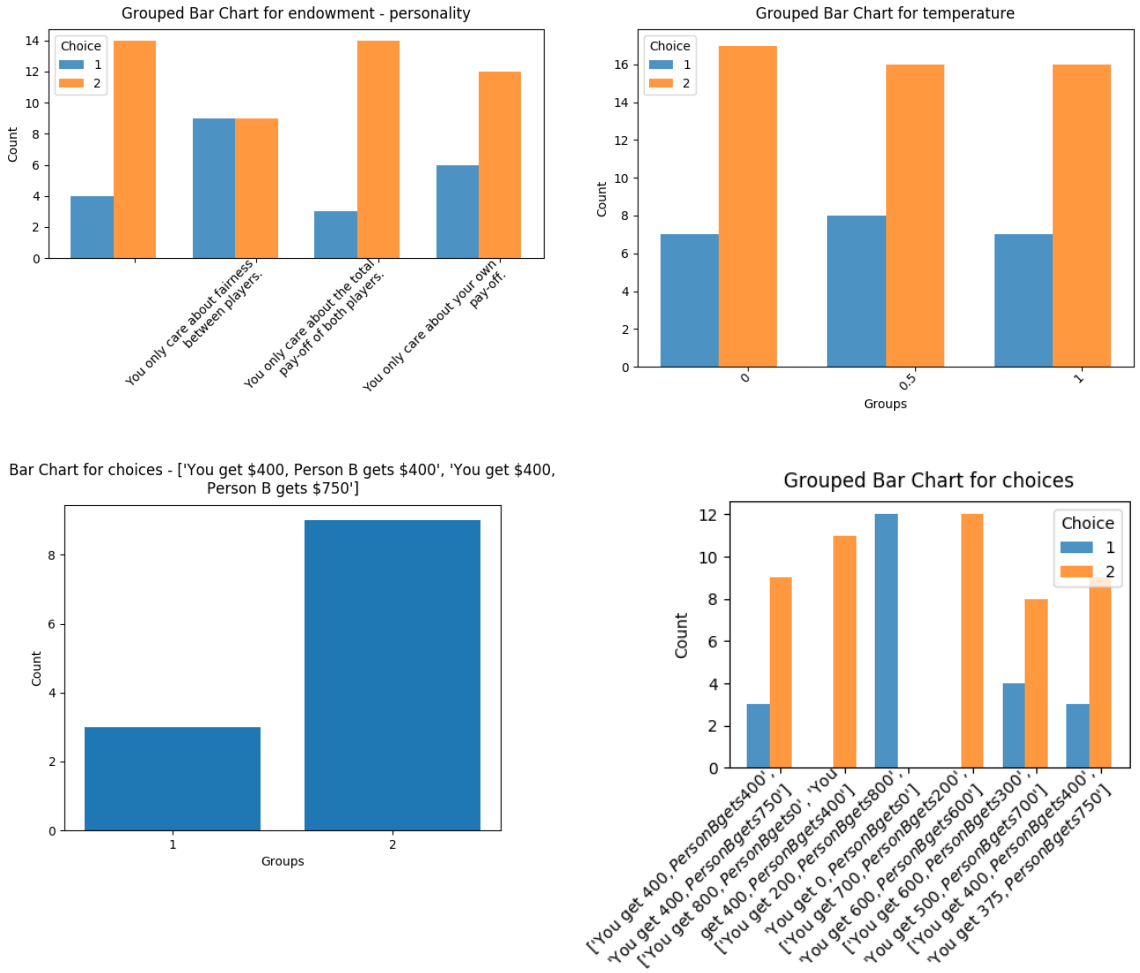

Figure 3-11: Automatically generated bar charts analyzing results from Charness and Rabin replication experiments.

to dive deeper. Because the charts are automatically built, some of the formatting may not be as intuitive or user-friendly as a custom chart — this is shown by the bottom-right chart in Figure 3-11 which automatically uses the very long choice text as the x-tick labels. However, the raw data is still available for researchers to analyze.

3.3.4 Flight Preferences Survey

Green and Wind described an outline for comparing user preferences regarding flight attributes using a conjoint analysis [24]. Varying several factors such as flight punctuality, airline, and number of stops, amongst other factors, they were able to compare

how each attribute contributed to consumer preferences.

We were interested in analyzing how GPT-3 would respond to experiments testing similar attributes through both a `Survey` and a `Conjoint`. We first tested a simple `Survey` comparing preferences between two flights, using attributes found in the Green and Wind conjoint. The dependent variables were the endowments, which varied age and attitudes towards flying. Choice 1 had 2/5 generally less preferred attributes while Choice 2 had 3/5 generally less preferred attributes based on Green and Wind's results, so Choice 2 was overall less preferred but more convenient. An example prompt is given below.

```
You are 40 years old.
You are taking a jet plane for a business appointment in Paris.
Which flight will you choose?

Your choices are the following:
1) A flight that is often late in arriving in Paris. The flight is
   nonstop, and it is anticipated that it will be half full. Flight
   attendants are warm and friendly and you would have a choice of two
   movies for entertainment.
2) A flight that is almost never late in arriving in Paris. The plane
   will make two intermediate stops, and it is anticipated that the
   plane will be mostly full. Flight attendants are cold and curt and a
   full entertainment system is provided.

What is your choice, with one word: [1, 2]:
```

The data storage and analysis is once again automatically completed. Example figures created in this pipeline are shown in Figure 3-12. The auto-generated bar charts immediately show potential trends in flight preference related to both dimensions of the endowment — age and flight preference.

Next, we used a conjoint analysis to identify which factors played a larger role in influencing decisions.

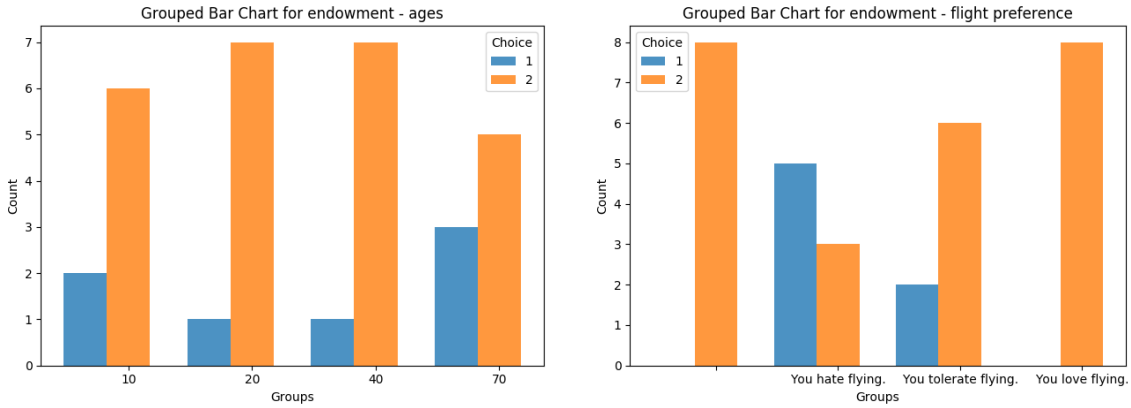

Figure 3-12: Automatically generated bar charts analyzing results from flight survey.

3.3.5 Replicating Green and Wind, Flight Preferences Conjoint

From the Green and Wind conjoint analysis, we chose a subset of factors to test [24]. We specifically used flight punctuality, passenger load, number of stops, flight attendant attitudes, and available entertainment.

```
You are a 40 year old person.

Which flight will you choose?

Your choices are the following:
1) This flight is never late, nonstop, anticipated to be half full, with
    warm and friendly flight attendants, and a choice of several movies.
2) This flight is never late, with 1 stop, anticipated to be half full,
    with warm and friendly flight attendants, and only one movie.

What is your choice, with one word: [1, 2]:
```

We randomized the order of choice sets, and ran the resulting pairs. A benefit of using AI subjects here is that every possible combination may be tested, which would be too many experiments to feasibly run using human subjects. However, we chose

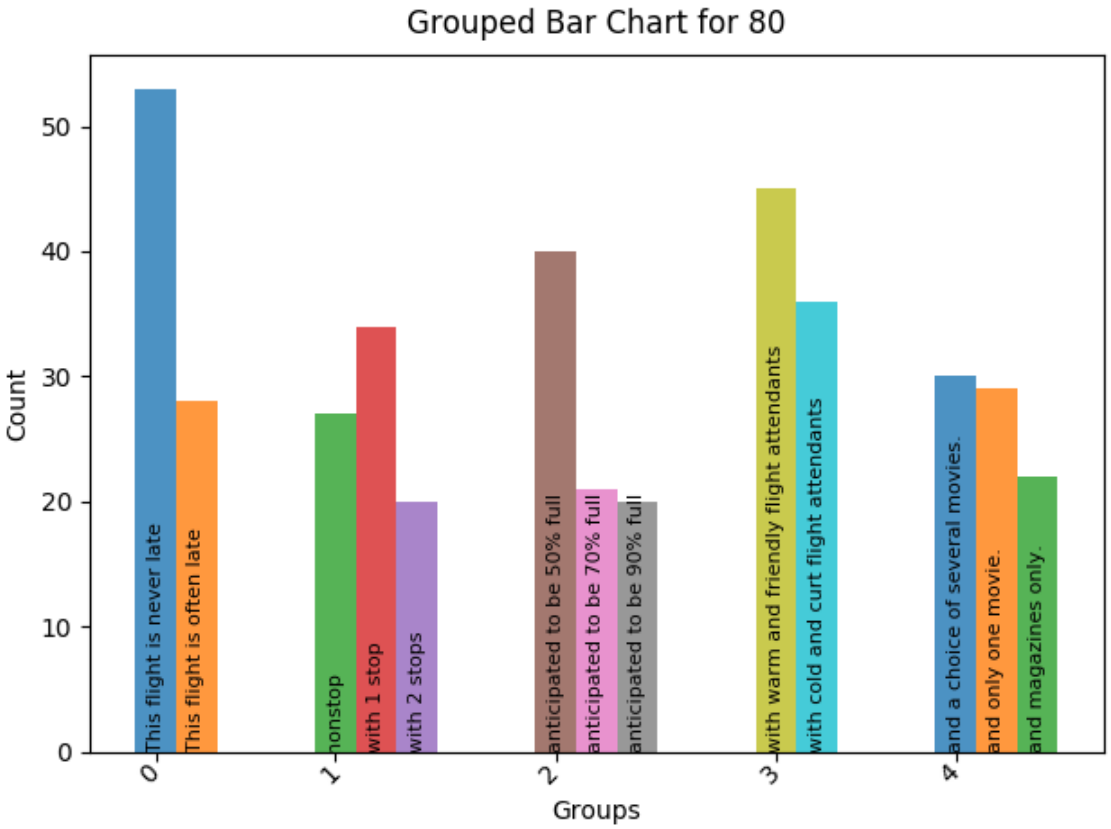

Figure 3-13: Automatically generated bar charts analyzing results from flight conjoint.

to stop after the first 100 results from our randomized list in order to mirror how the original experiment could only run a selection of prompts. The results are displayed in Figure 3-13.

We find trends very similar to those described by Green and Wind. The punctuality of the flight and attitude of the flight attendant created the most significant differences in results. The only outlier is that the number of stops played less of a role than in the original experiment, with the "1 stop" attribute being the majority in the chosen choices. This may also be the result of modernization in the flight industry from 1975 to 2020, where stops en route are less of an inconvenience now.

Generally, the chart is rough and difficult to fully personalize when automatically generated. However, researchers may certainly be able to glean general trends that can encourage them to dive deeper, without having to manually create and run the vast

number of potential combinations. Especially with experiments combining more types of attributes, using `homo silicus` to simulate an overview of the data would save immeasurable time, money, and effort in running a similar, large-scale experiment.

Chapter 4

Conclusion

In this thesis, we discussed the development of a Python-based library for using large language models as simulated subjects for economic experiments. By providing a framework for researchers to design, test, and analyze how AI subjects respond to various prompts, `homo silicus` provides a method for running experiments *in silico*. In developing an extensive and flexible pipeline, our library allows experimenters to immediately and cheaply iterate on potential projects without spending large amounts of money or having to recruit for specific audiences.

In our MVP, we designed a simple but flexible base set of modules capable of producing a variety of prompts and experiment structures. We tested this version by trying to replicate classical economic experiments and comparing outputs to those found by Horton [12]. Additionally, we experimented with manipulating other factors like prompt order, choice orders, and temperatures in order to understand impacts on results.

In the V2, we built on these modules by adding specialized classes, a method for serializing and sharing experiments, and post-processing results, along with miscellaneous additional functionalities. V2 is currently still under development, but the currently implemented features highly improve the usability of `homo silicus`. As we continue to iterate on the library, we will focus on supporting more types of experiments through the specialized classes, as well as testing the impacts of our additional functionalities.

4.1 Anticipated Uses and Viability

Our goal in developing this package was primarily to expose LLMs to economists as a potential experimental tool in an intuitive way. By handling the technical aspects of quickly combining prompts and running them using various APIs, our goal was to allow researchers to simply add the relevant text for their experimental prompts and then quickly get responses back that highlight interesting trends or potential results. Importantly, our purpose is to provide a tool for understanding potential human behavior in response to given scenarios, not to provide a replacement for empirical research altogether.

The code for a general experiment, similar to those run in the MVP sections (Section 3.2), would require a learning curve as it requires researchers to both format the text needed to general prompts and also to manually create `Experiment`, `Model`, `Endowments`, and `Round` objects, amongst others. Thus, users would have to know what each module represents and how to piece them together in order to run the experiments they need. Responses at the MVP stage were outputted as text strings so users would have to manually parse them to retrieve results.

In the V2, the addition of specialized classes allow the supported types of experiments to be run much more intuitively and with significantly less code and knowledge about the back-end of the package. Results can be written into real databases for permanent and parse-able storage, and there is support to automatically build data visualizations to provide experimenters with an overview of outcomes. Experiments can also be shared and run from JSON files, which would greatly simplify the process for replicating experiments on different machines or by different users.

4.2 Ethical Considerations

In this work, we used LLMs as simulated human agents in economic and social experiments. Although we found that they "behaved" similar to expectations of human behavior, there are significant ethical considerations before using them as direct rep-

resentations or replacements for people.

A significant benefit of using LLMs to create such a benchmark is that prompts can be run quickly and cheaply, and that biases in selecting an audience are eliminated. We can run individual prompts on new "people" each time, ensuring "fresh eyes" for each run (unlike some economic experiments that must run similar prompts on the same people to keep time and costs down). Additionally, there is no need to secure the same type of prior approvals necessary as with experiments run using human participants which allows these tests to be run immediately and changed easily [12].

However, there are several considerations before assuming LLMs fully represent the range of human behavior. Aher et al. points out that LLMs that do not inherently have certain different attributes may not be able to simulate differences between those attributes, regardless of endowment [2]. For example, a model with no gender differences cannot simulate the gender differences observed in traditional psychology studies.

Additionally, biases in LLMs are well-documented. Arising from biases in the training datasets themselves, these can reflect flaws in how that data was collected or biases present in society itself. Perpetuating these biases through the use of potentially prejudiced models would be incredibly harmful, and it is crucial to fully understand present biases in order to better interpret results.

Although the benefits of LLMs are many, measure the harms of language models is difficult and depends on how results are used [17]. If a specifically biased model is used unknowingly by a user of [**?**], this perpetuates toxicity and prejudice and skews results. Given the large impact even small perturbations in a prompt has on responses, this may also lead to errors or results inconsistent with general human behavior.

We would like to reiterate that this package is intended to be a tool in refining experimental design or to test experiments that may be too difficult to implement for human participants, but it is not a replacement for empirical research.

4.3 Future Work

We will continue to iterate on `homo silicus`, especially in creating new specialized classes, developing more of the functions described in Section 3.3.1, and building in more error handling across the package. As LLMs develop, `homo silicus` will have greater applications to a multitude of fields and experiments.

Appendix A

Appendix

I would like to acknowledge the use of ChatGPT in composing this thesis.

A.1 Prompts

A.1.1 Kahneman et al. Prompts

Prompts for various Kahneman et al. replication tests. Top: Horton (2023), Middle: General MVP prompt, Bottom: Manually improved MVP prompt

```
A hardware store has been selling snow shovels for $15. The morning after
    a large snowstorm, the store raises the price to $100.

Please rate this action as:
1) Completely Fair
2) Acceptable
3) Unfair
4) Very Unfair

You are a conservative.
What is your choice [1, 2, 3, or 4]:
```

```
A hardware store has been selling snow shovels for $15. The morning after
    a large snowstorm, the store raises the price to $100.

Please rate this action as:
0) Completely Fair
1) Acceptable
2) Unfair
3) Very Unfair

You are a conservative person named Subject 5.
What is your choice, with one word: [0, 1, 2, 3]:
```

```
A hardware store has been selling snow shovels for $15. The morning after
    a large snowstorm, the store raises the price to $100.

Please rate this action as:
1) Completely Fair
2) Acceptable
3) Unfair
4) Very Unfair

You are a conservative person.
What is your choice, with one word: [1, 2, 3, 4]:
```

A.1.2 Samuelson and Zeckhauser Prompts

Prompts for Samuelson and Zeckhauser experiment replications. Top: Horton [12], Bottom: `homo silicus`

The National Highway Safety Commission is deciding how to allocate its budget between two safety research programs: i) improving automobile safety (bumpers, body, gas tank configurations, seatbelts) and ii) improving the safety of interstate highways (guard rails, grading, highway interchanges, and implementing selective reduced speed limits).

The current budget allocation is 60\% *to auto safety and 40% to highway safety.*

They are considering the following options:

a) Decrease the highway program by 10% *of budget and raise the auto program by like amount*
b) Decrease auto program by 30% *of budget and raise the highway program by like amount*
c) Maintain present budget amounts for the programs
d) Decrease auto program by 10% *of budget and raise the highway program by like amount*

Your own views: highway safety and car safety are equally important

Please choose your most preferred option in light of your views ['a', 'b', 'c', 'd']:

Your view: highway safety and car safety are equally important
The National Highway Safety Commission is deciding how to allocate its budget between two safety research programs: i) improving automobile safety (bumpers, body, gas tank configurations, seatbelts) and ii) improving the safety of interstate highways (guard rails, grading, highway interchanges, and implementing selective reduced speed limits).
The current budget allocation is 60*% to auto safety and 40% to highway safety.*

They are considering the following options:
1) Decrease the highway program by 10*% of budget and raise the auto program by like amount*
2) Decrease auto program by 30*% of budget and raise the highway program by like amount*
3) Maintain present budget amounts for the programs
4) Decrease auto program by 10*% of budget and raise the highway program by like amount*

What is your choice: [1, 2, 3, 4]:

A.1.3 Charness and Rabin Prompt Orderings

Various prompt orderings for Charness and Rabin experiments run by `homo silicus`.
Top: Prompt Order 1, Middle: Prompt Order 2, Bottom: Prompt Order 3

```
You are a person named Subject 3. You only care about your own pay-off.
You are deciding on allocation for yourself and another person, Person A.

0) You get $400, Person A gets $400
1) You get $375, Person A gets $750

What is your choice, with one word: [0, 1]:
```

```
You are deciding on allocation for yourself and another person, Person A.

0) You get $400, Person A gets $400
1) You get $375, Person A gets $750

You are a person named Subject 3. You only care about your own pay-off.

What is your choice, with one word: [0, 1]:
```

```
You are deciding on allocation for yourself and another person, Person A.

You are a person named Subject 3. You only care about your own pay-off.

0) You get $400, Person A gets $400
1) You get $375, Person A gets $750

What is your choice, with one word: [0, 1]:
```

A.2 Code Samples

A.2.1 Charness and Rabin Replication Code

Charness and Rabin MVP Replication Code

```
from experiment.experiment import Experiment
from experiment.round import Round
from model import Model
from subject.endowments import Endowments

# Additional parameters for subject endowment
endowment_parameters = [
    "",
    "You only care about fairness between players.",
    "You only care about the total pay-off of both players.",
    "You only care about your own pay-off.", ]

# Outline main task
task = "You are deciding on allocation for yourself and
    another person, Person B."

# Generate choices for each scenario
scenarios = dict({
    "Berk29": ((400, 400), (750, 400)),
    "Berk26": ((0, 800), (400, 400)),
    "Berk23": ((800, 200), (0, 0)),
    "Berk15": ((200, 700), (600, 600)),
    "Barc8": ((300, 600), (700, 500)),
    "Barc2": ((400, 400), (750, 375)),
})
all_choices = []
for _, s in scenarios.items():
    all_choices.append([
        f"""You get ${s[0][1]}, Person B gets ${s[0][0]}""",
        f"""You get ${s[1][1]}, Person B gets ${s[1][0]}"""])
```

```
31
32 # Used for holding results
33 finals = [[]]
34
35 def run_experiment_charness_rabin():
36     model = Model()
37     experiment = Experiment()
38
39     for c in all_choices:
40         experiment.add_round(task=task, choices=c)
41
42     endowments = Endowments()
43     endowments.set_parameter(parameters=endowment_parameters,
44                              description="Personality",
45                              name="personality")
46
47     experiment.add_subjects(endowments)
48
49     prompts = experiment.generate_experiment_prompt()
50
51     # Run prompts and print corresponding results
52     for p in prompts:
53         print(p)
54         print([i["choice_text"] for i in model.run_prompt(p)])
55         print("------------------------------------------------")
56         if len(finals[-1]) >= len(scenarios):
57             finals.append([])
58         finals[-1].extend([i["choice_text"]
59             for i in model.run_prompt(p)])
60
61     print(finals)
```

Charness and Rabin V2 Replication Code

```
from classes.survey import Surveys
from model import Model
from subject.endowments import Endowments

# Set model
model = Model()

# Additional parameters for subject endowment
endowment_parameters = [
    "",
    "You only care about fairness between players.",
    "You only care about the total pay-off of both players.",
    "You only care about your own pay-off.", ]

endowments = Endowments()
endowments.set_parameter(parameters=endowment_parameters,
                         description="Personality",
                         name="personality")

# Outline main task
task = "You are deciding on allocation for yourself and
    another person, Person B."

# Generate choices for each scenario
scenarios = dict({
    "Berk29": ((400, 400), (750, 400)),
    "Berk26": ((0, 800), (400, 400)),
    "Berk23": ((800, 200), (0, 0)),
    "Berk15": ((200, 700), (600, 600)),
    "Barc8": ((300, 600), (700, 500)),
    "Barc2": ((400, 400), (750, 375)),
})
all_choices = []
```

```
for _, s in scenarios.items():
    all_choices.append([
        f"""You get ${s[0][1]}, Person B gets ${s[0][0]}""",
        f"""You get ${s[1][1]}, Person B gets ${s[1][0]}"""])


def run_experiment_charness_rabin():
    survey = Surveys([model], endowments, tasks=task,
        choices=all_choices, one_word=True, temperatures=[0],
        logprobs=3)

    print(survey.run_all())
```

Charness and Rabin V2 Replication from JSON Code

```
from classes.survey import Surveys

def run_experiment_charness_rabin():
    survey = Surveys.from_json(json_file_name)
    results = survey.run_all_write_data(additional_tables=
        ["endowment", "temperature", "choices"], analyze=True)
```

Bibliography

[1] OpenAI API.

[2] Gati Aher, Rosa I. Arriaga, and Adam Tauman Kalai. Using Large Language Models to Simulate Multiple Humans and Replicate Human Subject Studies, February 2023. arXiv:2208.10264 [cs] version: 3.

[3] Abdullah Almaatouq, Joshua Becker, James P. Houghton, Nicolas Paton, Duncan J. Watts, and Mark E. Whiting. Empirica: a virtual lab for high-throughput macro-level experiments. *Behavior Research Methods*, 53(5):2158–2171, October 2021.

[4] Lisa P. Argyle, Ethan C. Busby, Nancy Fulda, Joshua Gubler, Christopher Rytting, and David Wingate. Out of One, Many: Using Language Models to Simulate Human Samples. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 819–862, 2022.

[5] Marcel Binz and Eric Schulz. Using cognitive psychology to understand GPT-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120, February 2023.

[6] James Brand, Ayelet Israeli, and Donald Ngwe. Using GPT for Market Research. *SSRN Electronic Journal*, 2023.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[8] Gary Charness and Matthew Rabin. Understanding Social Preferences with Simple Tests. *The Quarterly Journal of Economics*, 117(3):817–869, 2002.

[9] Daniel L. Chen, Martin Schonger, and Chris Wickens. oTree—An open-source platform for laboratory, online, and field experiments. *Journal of Behavioral and Experimental Finance*, 9:88–97, March 2016.

[10] Marcus Giamattei, Kyanoush Seyed Yahosseini, Simon Gächter, and Lucas Molleman. LIONESS Lab: a free web-based platform for conducting interactive experiments online. *Journal of the Economic Science Association*, 6(1):95–111, June 2020.

[11] Paul E. Green and V. Srinivasan. Conjoint Analysis in Consumer Research: Issues and Outlook. *Journal of Consumer Research*, 5(2):103–123, 1978. Publisher: Oxford University Press.

[12] John J Horton. Large Language Models as Simulated Economic Agents: What Can We Learn from Homo Silicus?

[13] Daniel Kahneman, Jack L. Knetsch, and Richard Thaler. Fairness as a Constraint on Profit Seeking: Entitlements in the Market. *The American Economic Review*, 76(4):728–741, 1986.

[14] Richard A. et al. Klein. Many Labs 2: Investigating Variation in Replicability Across Samples and Settings. *Advances in Methods and Practices in Psychological Science*, 1(4):443–490, December 2018.

[15] R. Duncan Luce and John W. Tukey. Simultaneous conjoint measurement: A new type of fundamental measurement. *Journal of Mathematical Psychology*, 1(1):1–27, January 1964.

[16] Conor Mayo-Wilson and Kevin J. S. Zollman. The computational philosophy: simulation as a core philosophical method. *Synthese*, 199(1), December 2021.

[17] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, March 2022. arXiv:2203.02155 [cs].

[18] Peter S. Park, Philipp Schoenegger, and Chongyang Zhu. "Correct answers" from the psychology of artificial intelligence, March 2023. arXiv:2302.07267 [cs] version: 3.

[19] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training.

[20] William Samuelson and Richard Zeckhauser. Status quo bias in decision making. *Journal of Risk and Uncertainty*, 1(1):7–59, March 1988.

[21] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M. Rush. Interactive and Visual Prompt Engineering for Ad-hoc Task Adaptation with Large Language Models. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1146–1156, January 2023.

[22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2023. arXiv:2201.11903 [cs].

[23] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT, February 2023. arXiv:2302.11382 [cs].

[24] Jerry Wind and Paul Green. A New Way to Measure Consumer Judgments. *Harvard business review*, 53:107–117, July 1975.

[25] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate Before Use: Improving Few-Shot Performance of Language Models, June 2021. arXiv:2102.09690 [cs].