Evaluating the Decency and Consistency of Data Validation Tests Generated by LLMs*

An application to Canadian political donations data

Rohan Alexander[†] Lindsay Katz[‡] Callandra Moore[§]

Michaela Drouillard[¶] Michael Wing-Cheung Wong[‖]

Zane Schwartz[**]

April 1, 2024

We investigated whether large language models (LLMs) can develop data validation tests. We considered 96 conditions each for both GPT-3.5 and GPT-4, examining different prompt scenarios, learning modes, temperature settings, and roles. The prompt scenarios were: 1) Asking for expectations, 2) Asking for expectations with a given context, 3) Asking for expectations after requesting a data simulation, and 4) Asking for expectations with a provided data sample. The learning modes were: 1) zero-shot, 2) one-shot, and 3) few-shot learning. We also tested four temperature settings: 0, 0.4, 0.6, and 1. And the two distinct roles were: 1) helpful assistant, 2) expert data scientist. To gauge consistency, every setup was tested five times. The LLM-generated responses were benchmarked against a gold standard data validation suite, created by an experienced data scientist knowledgeable about the data in question. We find there are considerable returns to the use of few-shot learning, and that the more explicit the data setting can be the better,

*We thank Monica Alexander, Chris Maddison, Dianne Cook, Yadong Liu, and Will Jettinghoff for helpful suggestions. Code and data: https://github.com/RohanAlexander/evaluating_decency_and_consistency. Contact: rohan.alexander@utoronto.ca. Contributions: RA had the initial idea. RA and CM developed the experimental set-up. ZS established the political donations datasets and obtained funding. LK developed the initial suite of dataset validation tests that we compare the LLMs against. RA wrote the code to interact with the LLMs and did the modelling. RA, MD, and MW evaluated the LLM output. All authors contributed to writing the initial draft as well as improving and finalizing the paper.

[†]Information and Statistical Sciences, University of Toronto. rohan.alexander@utoronto.ca.

[‡]The Investigative Journalism Foundation

[§]The Investigative Journalism Foundation

[¶]The Investigative Journalism Foundation

[‖]Massachusetts Institute of Technology

[**]The Investigative Journalism Foundation

to a point. The best LLM configurations complement, rather than substitute, the gold standard results. This study underscores the value LLMs can bring to the data cleaning and preparation stages of the data science workflow, but highlights that they need considerable evaluation by experienced analysts.

1 Introduction

The Investigative Journalism Foundation (IJF) created and maintains a dataset related to political donations in Canada. As of September 2023, the dataset comprised 9,204,112 observations and 15 variables. Every day new observations are added, based on newly released donations records made available by provincial and federal elections agencies. This release cycle is variable, with periods of inactivity followed by bursts of multiple releases. Katz and Moore (2023) manually construct an extensive suite of automated tests for this dataset. These impose certain minimum standards on the dataset, including: that constituent aspects add to match any total, class is appropriate, and null values are as expected. This suite allows researchers to use the dataset with confidence and ensures that new additions are fit for purpose.

We revisit this suite of tests to determine whether we can use large language models (LLMs) to mimic this suite of validation tests. Our estimand is the similarity between those tests written by an LLM and those written by an experienced data scientist. We consider a variety of prompts, roles, learning modes, and temperature settings, resulting in an experiment with 96 conditions. In particular, we consider four prompt variations: ask for expectations; ask for expectations given described context; ask for expectations having first asked for a data simulation; and ask for expectations given a sample of data. We also consider zero-, one-, and few-shot learning; four temperature values: 0, 0.4, 0.6, and 1 (which should influence how unexpected the results are); and two roles: helpful assistant and expert data scientist. For every combination we obtain five responses from the LLM. We run all conditions separately for both GPT-3.5 and GPT-4.

Three human coders judge the responses produced by the LLMs, rating their decency (1-5, where 1 is the worst and 5 is the best) and their consistency (1-5, where 1 means they are very different and 5 means they are essentially identical). We then build an ordinal regression model with `rstanarm` to explore the relationships that decency and consistency have with prompts, roles, learning modes, and temperature settings.

We find there are considerable returns to one- and few-shot learning. We also find that including detailed descriptions of the expected dataset can help improve the quality of the tests that are produced, even more than including an example of the dataset. Temperature is associated with consistency, but not with decency. Surprisingly, there was not much of a difference found between GPT-3.5 and GPT-4.

Our results demonstrate one use for LLMs in production, outside of just analysis. In particular, data scientists are often concerned that results may be an artifact of some errors in the data cleaning and preparation pipeline. Data validation tests help assuage these concerns but they can be time consuming to produce. This paper shows it is possible to use LLMs to establish an initial suite of tests, which may encourage their use. But the variability of responses means these are best seen as a starting place.

The remainder of this paper is structured as follows: Section 2 provides a brief overview of the underlying dataset about which we are writing tests, as well as LLMs. Section 3 details the human coded LLM responses, which is the analysis dataset used in this paper. Section 4 specifies the analysis model used and Section 5 details the estimates and results. Finally, Section 6 discusses some of the implications of this study and details a few weaknesses.

2 Background

2.1 The political donations dataset

The Investigative Journalism Foundation (IJF) is a nonprofit news media outlet that is centered around public interest journalism. Their mandate is to help rebuild trust in Canadian democracy and hold the powerful accountable through data-driven investigative reporting. A core component of the IJF's work is building and actively maintaining nine publicly available databases with information on political donations, registered charities, and political lobbying in Canada. While the information that these databases are built upon are ostensibly public, they are typically not maintained or available in a way that is widely accessible or conducive to analysis. Further, the format and accessibility of these data vary over time and across jurisdictions, making it difficult to look at temporal or regional differences in the data. The IJF's collation of these databases, and the subsequent high-quality journalism informed by these data, serves as a crucial contribution to rebuilding trust and transparency in Canadian democracy.

One of the IJFs eight databases, and the focus of this work, is the political donations database. Canadian legislation requires political parties and candidates to disclose records of financial contributions they receive. These records are maintained by elections agencies across provinces and territories, and at the federal level. The frequency and scope of these disclosures vary across jurisdictions. For instance, in British Columbia, parties, candidates, constituency associations, and leadership and nomination contestants can receive political donations, while in Newfoundland and Labrador, donation recipients are limited to only parties and candidates (The IJF 2023). The IJF's political donations database is a compilation of these political finance records across all Canadian jurisdictions, with data spanning from 1993 to the present day. The database contains 15 variables including the donor's name, the political party and entity to which the donation was made, the amount donated, as well as the region and year of the donation.

While the IJF's database is available in a clean, user-friendly format, the original records upon which it was created were not all accessible in this way. The format of donation records varies across jurisdiction and time. While some are available in readily downloadable spreadsheets, others are available as PDF or HTML files—the former necessitating the use of optical character recognition (OCR) (The IJF 2023). To prepare their database for publication, the IJF team performed significant manual cleaning. The majority of this work resulted from the conversion of PDF donation records to rectangular CSV format using OCR. This is prone to scanning-related errors, such as the number 0 being scanned as the letter O. The IJF manually corrected these errors wherever they were identified by comparing the machine-legible OCR output to the original PDF donations record (The IJF 2023).

Additional cleaning was done for the purpose of data legibility. For instance, the IJF standardized donation dates to match the YYYY-MM-DD format, and they standardized donor names which were in the format "Surname, First name" to be in the form of "First name Surname" (The IJF 2023). Party names and donor types were also standardized for consistency, and donation records with an abbreviated party name were supplied with the complete name for that party. Finally, in rows where the donor type was null but only individuals were legally allowed to make donations in that jurisdiction and year, the IJF changed that null entry to be "Individual" (The IJF 2023).

Despite the cleaning performed by the IJF team, the magnitude of these data coupled with their self-reported nature makes them prone to both human and computational error arising from the parsing process. With over 9.2 million rows in this database, it would be difficult for the IJF to manually identify all errors and inconsistencies present in their data—whether that is as minor as an incorrectly formatted date, or as major as a reported donation amount thousands of dollars above the legal limit. To address this challenge, the IJF uses computational tools to assess data quality (Katz and Moore 2023). The team has developed a suite of comprehensive data tests for all nine databases using Python's `Great Expectations` library. This contains a number of pre-defined functions to test that particular characteristics expected of a dataset hold true in the data at large.

As described by Katz and Moore (2023), the process of developing this test suite was iterative and necessitated significant domain knowledge to develop accurate and valuable tests. For the political donations database, Katz and Moore (2023) developed a suite of tests pertaining to missingness, formatting, and value expectations in the data. For instance, they set the expectation that for donations made in British Columbia, Ontario, or federally, the `donation_date` entry should not be missing, because those three jurisdictions collect that variable (Katz and Moore 2023). They also test that, say, donations data from 2022 align with the 2022 legal donation limits for each jurisdiction, and that, where applicable, the sum of reported monetary and non-monetary contribution amounts add up to the total reported donation amount. Katz and Moore (2023) provide further details on the development and implementation of these tests.

2.2 Large language models

Historically, natural language processing (NLP) models operated in smaller contexts than today. The first NLP systems were rule-based, relying on hand-crafted linguistic rules based on experts' understanding of languages, and were trained on much smaller datasets.

By the 1990s, advancement in computational capabilities and an increased availability in larger textual datasets allowed for a paradigm shift from rule-based systems to statistical methods. Models learned patterns from data rather than relying on manual rules, and probabilistic models such as Hidden Markov Models (HMMs) and Bayesian networks underpinned new NLP applications. Where rule-based models were brittle and complex, with existing functionalities breaking with the addition of new rules, probabilistic models inferred linguistic patterns from large datasets, which made them better at scaling and generalizing to solve unseen linguistic challenges.

The transition from statistical methods and traditional machine learning methods to "deep learning" came when Bengio, Ducharme, and Vincent (2000) introduced neural network based language models. These models predicted the next word in a sentence, given the previous words. The model of Bengio, Ducharme, and Vincent (2000) relied on a representation for words, where every word from the vocabulary is linked with a continuous vector. During training, models refined these vector representations. Words with similar meanings had vectors that were closer together in vector space, which enabled the model to understand semantic relationships between words.

Once neural networks were introduced, many significant advancements such as word embeddings (Pennington, Socher, and Manning 2014; Mikolov et al. 2013), long short-term memory (Hochreiter and Schmidhuber 1997) and Convolutional Neural Networks (CNNs) (Kim 2014) followed. The concept of pre-training models and then fine-tuning them for specific tasks, as seen with models like ULMFiT (Howard and Ruder 2018) and ELMo (Peters et al. 2018), started to take root in the late 2010s, which set the stage for transformer-based architectures.

Vaswani et al. (2017) introduced transformer architectures, which enabled models to understand complex linguistic patterns and generate text by processing vast amounts of text data. These models, which are characterized by parameter counts among billions or trillions, are trained on diverse corpora, covering cultural nuances and varieties of human knowledge. Operating with this broader context has led to LLMs in recent years setting new benchmarks across various NLP tasks, with a potential to reshape many industries.

LLMs are customized and refined through in-context learning using prompting (Ouyang et al. 2022). A prompt is a set of natural language instructions which define the parameters and context for the desired output and may include one or more input-output examples. By using this approach, LLMs can apply their existing knowledge gained from training on various datasets to adapt to new contexts specified by the prompt. The outcomes generated by

LLMs are sensitive to the specific phrasing and structure of these natural language instructions. Consequently, there is ongoing work to establish effective prompt patterns (White et al. 2023).

There has been some work using LLMs for data cleaning, including H. Zhang et al. (2023) who focus on data pre-processing.

3 Data

We are interested in the extent to which the LLMs can develop a suite of data validation tests that is similar to a suite developed by an experienced expert data scientist who is familiar with the dataset. To test this, we establish and run a series of experiments where we consider different specifications and then compare the LLM output. The variables that we consider are summarized here, and the full details are included in Appendix A.

- Four prompts:
  - List the variables of interest—"index", "amount", "donor_location", "donation_date", "donor_full_name", "donor_type", "political_entity", "political_party", "recipient", "region", "donation_year", "amount_monetary", "amount_non_monetary", "electoral_event", "electoral_district", "added"—and ask for a suite of expectations using the Python package `great_expectations`.
  - As above, plus provide details about each variable. For instance, "amount" is a monetary value that cannot be less than \$0. An example observation is "195.46". It is, possible, but unlikely to be more than \$1,000.00. It cannot be NA. It should be a numeric. The maximum donation "amount" depends on the value of "region" and "year". For "Federal" is 1675, for "Quebec" is 100 since 2013 and 500 for earlier years, for "British Columbia" is 1309.09, for "Ontario" is 3325, and for "Alberta" is 4300. There is no limit for "Saskatchewan".
  - As above, plus ask the LLM to simulate an example dataset of 1000 observations.
  - Provide the details of the dataset in the first option, as well as an extract of the dataset.
- Three learning modes: zero-, one-, and few-shot learning. In the first, we do not provide any examples; in the second, we provide one example; and in the third, we provide three examples.
- Four temperature values: 0, 0.4, 0.6, and 1.
- Two roles:
  - You are a helpful assistant.
  - You are a highly-trained, experienced, data scientist who is an expert at writing readable, correct, Python code.

This combination of variables and options results in 96 different prompt situations. We run these through both GPT-3.5 and GPT-4, using the OpenAI API. For every combination we ask for five responses to understand variation.

This results in a dataset of responses. The combination of settings that gave rise to each response was blinded and the order randomized, and then the responses were ranked by three experienced human evaluators on two metrics.

1. Human evaluators ranked the "decency" of the first response for each combination of variables. This is a measure of how effective the LLM validation tests were compared with the code written by the experienced data scientist who wrote the original suite of tests. The LLM responses are not expected to have the full context of the code, so we do not expect an exact match, but it should actually write code, import relevant libraries, add comments, deal with class, and write a variety of relevant expectations. 1 means that the code is unusable, 2 means that it is not unusable but would need a lot of work and would be disappointing from a human, 3 means that it is fine but would need some fixing, 4 means it is broadly equivalent to what the gold standard suite contains, and 5 means it is in no worse and is better in some way than the gold standard validation suite. The decency rating is at a condition level, focusing on only the first response given that combination of settings. We consider only the first given constraints on the evaluators' time.
2. The other, "consistency", is a ranking 1-5 of how different each of the five responses was for that particular combination of variables. 1 means that responses 1-5 were wildly different. 5 means that responses 1-5 are entirely or essentially the same.

To bring the three evaluators together we consider the median response. This then allows us to examine whether there is a relationship between decency and consistency (Figure 1). There is a broadly positive relationship. Details about coder-specific responses are provided in Appendix B.

Figure 2 illustrates how decency and consistency differ based on whether GPT-3.5 or GPT-4 is used. Unexpectedly, GPT-4 has fewer responses rated 5/5 for decency, compared with GPT-3.5 with 3% compared with 9% (Figure 2a). Overall, the mean decency response for GPT-3.5 is 3.26, while the mean decency of the responses generated by GPT-4 is 3.19. The consistency is not too different between the two versions, with GPT-3.5 having an average of 3.21, while GPT-4 has an average of 3.33. Unexpectedly, GPT-4 appears to have fewer responses that are essentially identical i.e. 5/5 (Figure 2b).

Figure 3 examines how decency and consistency differ based on which prompt is used. The prompts differ by how much information is provided. The least informative prompt, "Name", essentially consists of just providing the LLM with the names of the columns that are expected to be in the dataset. The next most informative prompt, "Describe", adds a detailed description of what we expect of the observations. "Simulate" adds that we expect the LLM to first simulate a dataset based on that description, before generating the expectations. And finally,

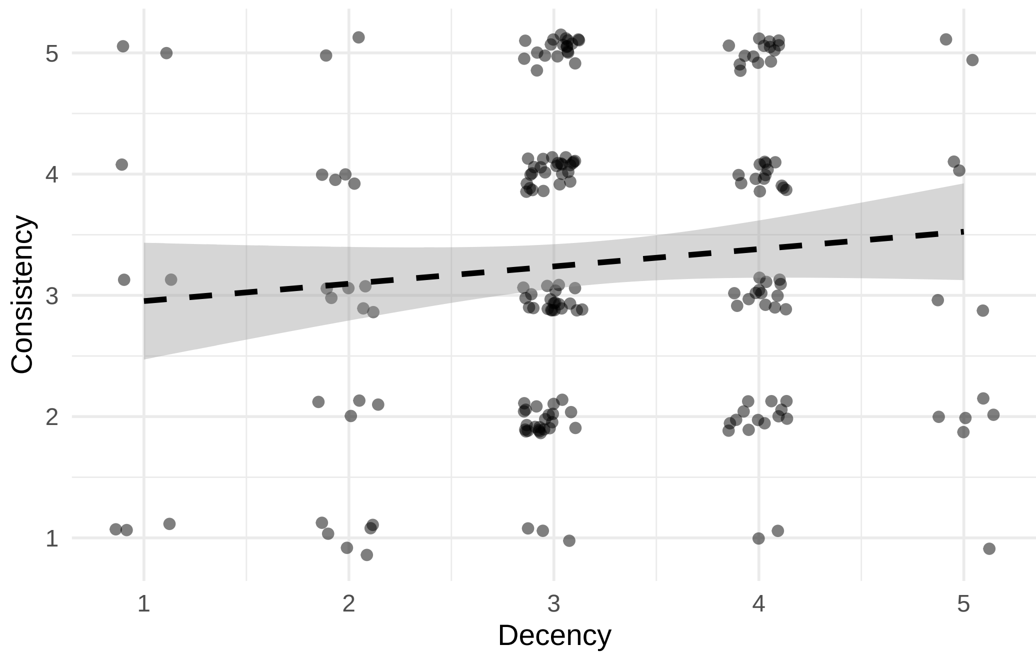
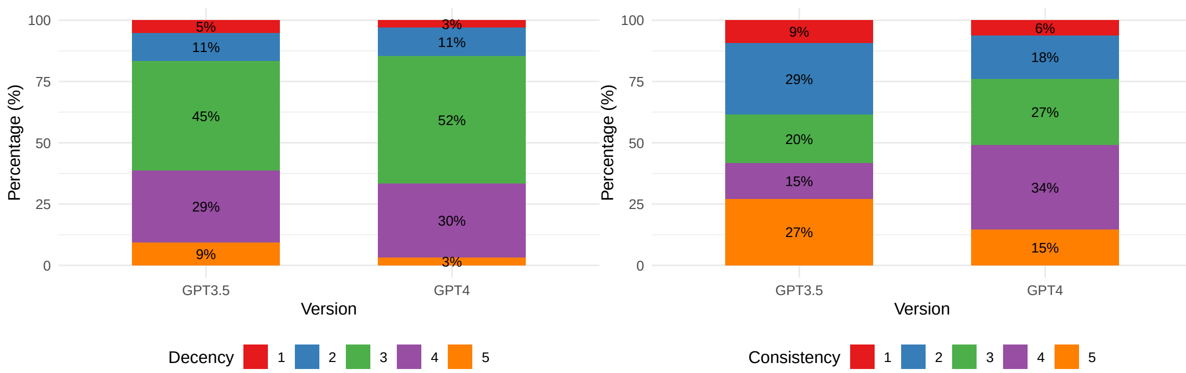


Figure 1: How decency and consistency compare with each other. Higher decency means the LLM responses correspond more closely with the gold standard. Higher consistency means all five responses were similar to each other.



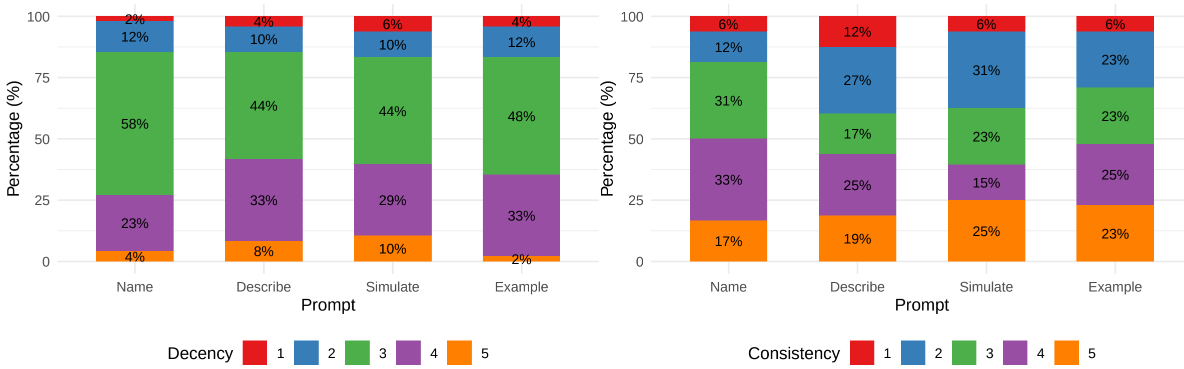
(a) Decency (b) Consistency

Figure 2: How decency and consistency change based on whether GPT-3.5 or GPT-4 is used. For decency, 1 is the worst and 5 is the best. For consistency 5 means all responses were essentially the same, while 1 means at least one was quite different.

the most informative prompt, "Example", provides a snapshot of the dataset, consisting of the relevant variables and ten observations.

There is a difference in terms of how the prompts are associated with decency (Figure 3a). In particular, "Name" is almost never associated with a 5/5 rating, and it has many responses rated 3/5. Surprisingly the most informative prompt, "Example", is also never associated with a 5/5 rating. Instead it is "Describe" and "Simulate" that tend to be associated with better decency ratings. This is reflected in the averages, which are 3.15, 3.31, 3.27, and 3.17 for "Name", "Describe", "Simulate", and "Example", respectively.

The pattern is less clear when it comes to consistency (Figure 3b). All four have similar averages, at 3.42, 3.10, 3.21, and 3.35 for "Name", "Describe", "Simulate", and "Example", respectively. A wider variety of responses (as denoted by lower consistency ratings), is rarely seen for "Name" and "Describe". It is surprising that "Name" should have the highest average.



(a) Decency

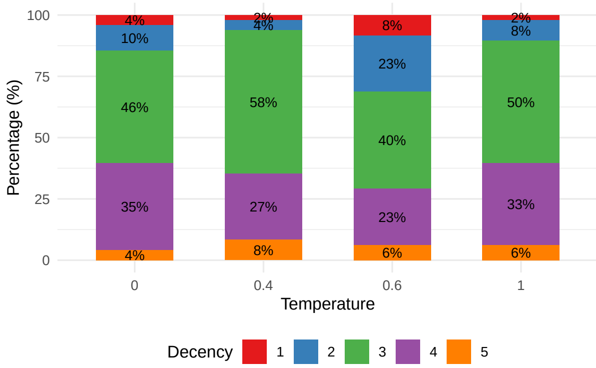
(b) Consistency

Figure 3: How decency and consistency change based on the type of prompt used. Name means essentially only the variables of interest are listed. Describe means some detail about each variable is provided. Simulate means the LLM is asked to first think about simulating the variables. Example means an extract of the dataset is provided.

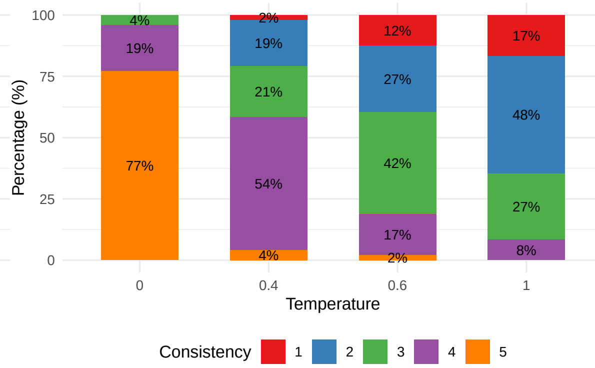
Temperature is a parameter that varies from 0 to 1, that we use to manipulate how random a LLM is. At high temperatures, an LLM will produce a wider variety of responses. At lower temperatures it will focus on the single most likely response. Higher temperature should be associated with a wider variety of LLM responses.

Figure 4 examines how decency and consistency differ based on which of the four temperature values we consider—0, 0.4, 0.6, 1—is used. There appears to be limited difference in terms of how different temperature values are associated with decency (Figure 4a). They all have similar mean values at 3.25, 3.35, 2.96, and 3.33 for temperature values of 0, 0.4, 0.6, and 1, respectively.

In contrast, a sizeable different in consistency can be seen with different temperatures values (Figure 4b). Temperature values of 0 are associated with ratings of high consistency, and higher temperatures are associated with lower consistency. Their means differ considerably, with 4.73, 3.40, 2.69, and 2.27 for temperature values of 0, 0.4, 0.6, and 1, respectively. The one unexpected aspect is temperature of 0.6, which has an outsized number of responses with decency of 1/5 or 2/5.



(a) Decency



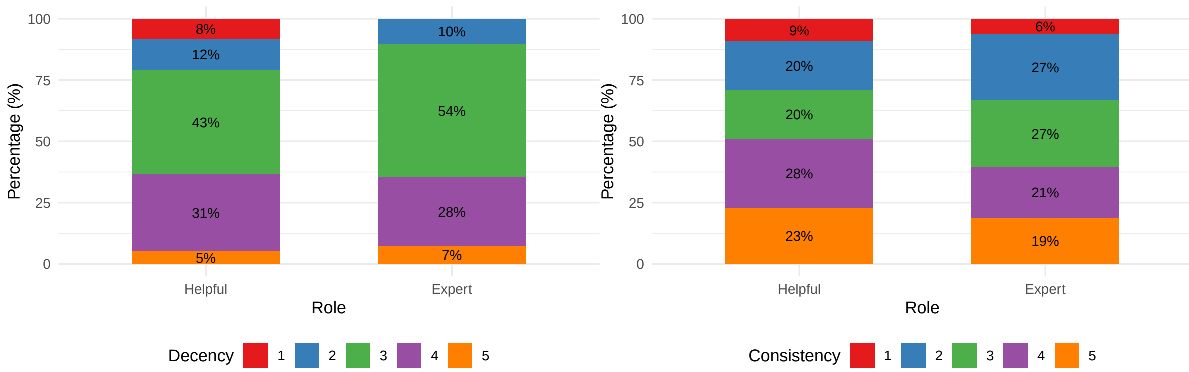
(b) Consistency

Figure 4: How decency and consistency change based on temperature. Temperature of 0 means that less variation is expected from the outputs, while temperature of 1 means considerable variation is expected.

Role is an aspect of a prompt that is provided to the LLM before the main prompt content. We considered two different roles, one that positioned the LLM as a helpful assistant, and the other that positioned the LLM as an experienced data scientist (Figure 5). We were expecting that the expert role would result in better code, but there was no obvious difference in terms of decency (Figure 5a) or consistency (Figure 5b). Although it is notable that there are no responses rated 1/5 with the "Expert" role. Their mean decency did not differ by much in either case with 3.12 and 3.32 for "Helpful" and "Expert", respectively, while mean consistency was 3.35 and 3.19.

Learning mode refers to the number of examples provided to the LLM as part of the prompt. Zero-shot learning means that no examples are provided, while one-shot and few-shot learning refer to one- and a few- examples being provided, respectively. Although the advantage of LLMs such as GPT-3.5 and GPT-4 is that they can do well with zero-shot learning, we would expect that they will do better with one-shot and few-shot.

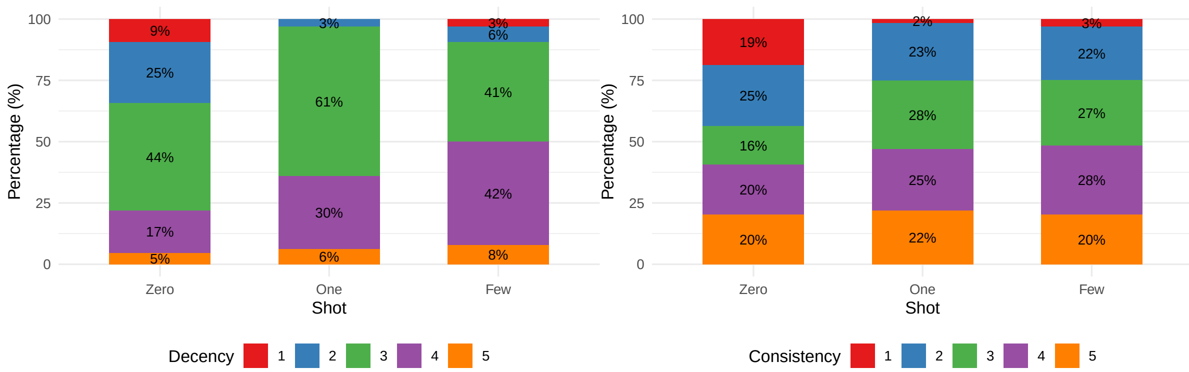
We see substantial differences, especially when moving away from zero-shot learning (Figure 6). In particular, we see that decency of 1/5 and 2/5 is dominated by zero-shot (Figure 6a). This is also reflected in the mean decency which for zero-shot is 2.83, while for one- and few-shot learning is 3.39 and 3.45, respectively. This is largely due to the relative reduction in 1/5 and 2/5.



(a) Decency (b) Consistency

Figure 5: How decency and consistency change based on whether role is 'Helpful' or 'Expert'. This is considered before the main prompt content and is designed to position the LLM as having a certain role.

We see this pattern in consistency as well. For instance, zero-shot learning is over-represented in the least consistent responses, both 1/5 and 2/5 (Figure 6b). And the mean level of consistency is lower for zero-shot learning, at 2.98, compared with single- and few-shot, at 3.42 and 3.41, respectively.



(a) Decency (b) Consistency

Figure 6: How decency and consistency change based on zero-, one-, and few-shot learning. This refers to the number of examples given to the LLM with zero-shot learning meaning no examples are provided, while few-shot learning means a few are given.

4 Model

Consistency and decency, our dependent variables, are ordered, categorical, outcomes. As such we model the rating Y, which is an ordinal outcome with $J = 5$ possible categories, using ordered logistic regression. Such a model works by estimating the probability that the outcome is less than or equal to some specific category, which in this case is the coded rank of that response (i.e. 1, 2, 3, 4, 5). Models for consistency and decency are estimated separately. In latent variable formulation, we assume that the observed category y can be related to an underlying continuous latent variable, y^*, through a series of cutpoints:

$$y = \begin{cases} 1 & \text{if } y^* < \zeta_1 \\ 2 & \text{if } \zeta_1 \leq y^* \\ \vdots & \\ J & \text{if } \zeta_{J-1} < 1 \end{cases}$$

where ζ is a vector of cutpoints. The latent variable y^* is then assumed to have a logistic distribution and is modeled as a linear function of covariates. In our case, we are interested in exploring the relationships that consistency and decency have with model, prompt, temperature, role, and learning mode:

$$y^* = \beta_1 \cdot \text{version}_i + \beta_2 \cdot \text{prompt}_i + \beta_3 \cdot \text{temperature}_i + \beta_4 \cdot \text{role}_i + \beta_5 \cdot \text{shot}_i$$

In terms of our predictors, version is a binary variable as to whether we are using GPT-3.5 or GPT-4. Before looking at the data, we expected that GPT-4 would do better in terms of both decency and consistency than GPT-3.5. Prompt can be one of four values: "Name", "Describe", "Simulate", and "Example". We expected that "Simulate" and "Example" would be associated with higher decency than "Name" and "Describe". However, we expected the opposite relationship with consistency. This is because we expected that what would increase decency would be the more specific guidance provided by the "Simulate" and "Example" prompts, which would also increase the consistency. Temperature can take one of four values: 0, 0.4, 0.6 and 1. We expected that higher temperature values would be associated with less consistency. However, it was difficult to anticipate how temperature should be related to decency. It could be that allowing more "randomness" enables better responses, or it could be that it allows more scope to be wrong. Role can be one of two values: "Helpful", or "Expert", while learning mode can be one of three: "zero", "one", or "few". In the case of both role and learning mode, we expected that the "Expert" role, and one- and few-shot learning would be associated with higher decency, and consistency.

We fit this model, separately, for each of consistency and decency, in a Bayesian framework using the package `rstanarm` (Goodrich et al. 2023) and the R statistical programming language (R Core Team 2023). This computational process requires priors to be placed on R^2, the proportion of variance in the outcome that is attributable to the coefficients in a linear model, and the vector of cutpoints ζ. For the prior on R^2, we follow Gelman, Hill, and Vehtari (2020, 276) and assume the mean is 0.3. For the vector cutpoints ζ, we use an uninformative prior of Dirichlet(1). Gabry and Goodrich (2020) provide more information about fitting this type of model using `rstanarm`. Model diagnostics are provided in Appendix C.

5 Results

5.1 Model-based results

In total we considered 192 observations, which is 96 for each GPT-3.5 and GPT-4. The estimates from our models are shown in Table 1 and Figure 7, which were both produced with `modelsummary` (Arel-Bundock 2022).

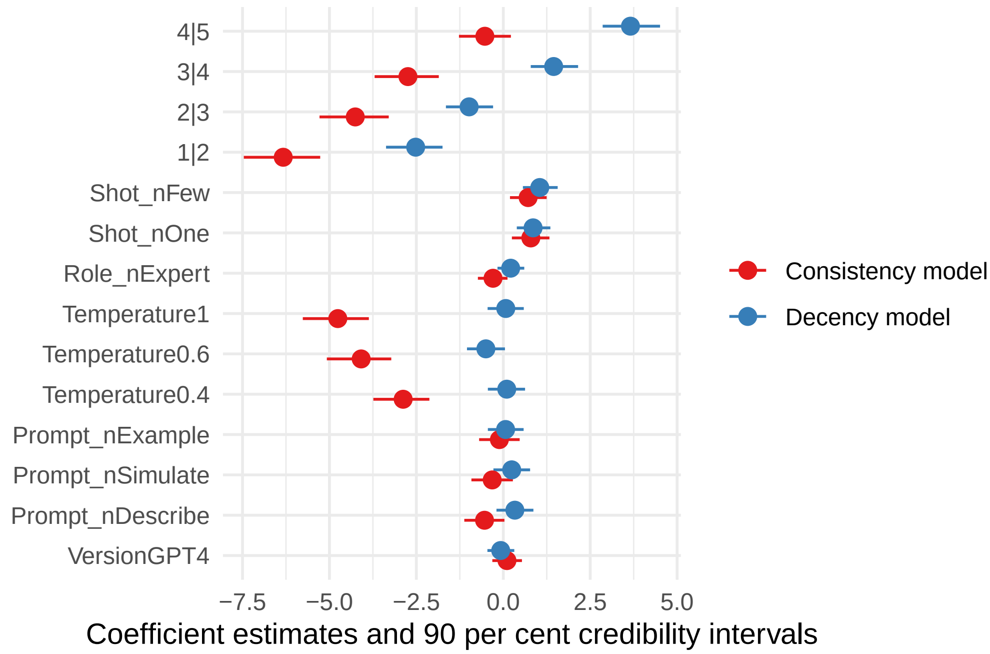


Figure 7: Exploring the relationships that consistency and decency have with model, prompt, temperature, role, and learning mode. The x-axis shows coefficient estimates and 90 per cent credibility intervals.

Table 1 shows the coefficient estimate with the mad (mean absolute deviation) statistic in brackets. In Figure 7, the dot represents the coefficient estimate, while the lines are 90 per cent credibility intervals.

The intercepts, 1|2, 2|3, 3|4, and 4|5, shown in Table 1 reflect cutpoints, where a rating goes from one category to the next (e.g. from 1 to 2, etc).

The models do not identify a substantial difference between GPT-3.5 and GPT-4 in terms of consistency, and, surprisingly, even a somewhat negative association in terms of decency.

They also do not identify much difference between the different prompt types in terms of consistency. However, they do find that "Describe" and "Simulate" are associated with increased decency, as is "Example", although surprisingly to a lesser extent.

Table 1: Exploring the relationships that consistency and decency have with model, prompt, temperature, role, and learning mode. The mad (mean absolute deviation) statistic is in brackets.

	Consistency model	Decency model
VersionGPT4	0.11	−0.07
	(0.25)	(0.23)
Prompt_nDescribe	−0.54	0.33
	(0.36)	(0.31)
Prompt_nSimulate	−0.32	0.24
	(0.35)	(0.33)
Prompt_nExample	−0.11	0.07
	(0.35)	(0.32)
Temperature0.4	−2.88	0.10
	(0.49)	(0.33)
Temperature0.6	−4.09	−0.50
	(0.55)	(0.33)
Temperature1	−4.76	0.07
	(0.58)	(0.32)
Role_nExpert	−0.30	0.21
	(0.26)	(0.24)
Shot_nOne	0.79	0.86
	(0.32)	(0.29)
Shot_nFew	0.71	1.05
	(0.32)	(0.30)
1|2	−6.33	−2.52
	(0.65)	(0.49)
2|3	−4.26	−0.98
	(0.60)	(0.42)
3|4	−2.74	1.45
	(0.56)	(0.42)
4|5	−0.53	3.66
	(0.47)	(0.49)
Num.Obs.	192	192
ELPD	−224.7	−240.3
ELPD s.e.	8.1	10.0
LOOIC	449.5	480.6
LOOIC s.e.	16.2	20.1
WAIC	449.4	480.6

As expected, increases in temperature are strongly associated with less consistency. However, we are unable to identify much of a relationship between temperature and decency.

The models struggle to find any association between which role is used in terms of either consistency or decency. That is, priming the prompt with either "You are a helpful assistant." or "You are a highly-trained, experienced, data scientist who is an expert at writing readable, correct, Python code." did not seem to change much in terms of the coding of the responses.

Finally, one- and few-shot learning is associated with increased decency and slightly increased consistency over zero-shot prompts. That said, the magnitude of this relationship is not overly large.

5.2 Coder-based results

The rating of the coherence and decency was a human endeavor, with all the bias that brings. One benefit is that the coders were able to leave comments about the LLM outputs as they rated them. The provides an opportunity for additional results, in addition to those based on their quantitative ratings.

In terms of decency, sometimes the LLMs simply did not generate code, instead generating natural language text about what tests could look like. Other times the LLMs generated a lot of very similar tests, for instance going through many possible options testing for class or non-null results, at the expense of developing a broadly useful suite. Finally, sometimes the LLMs would generate only one or two tests, or generate tests without any comments or other explanations.

The responses that "excelled" in their decency scores tended to bring a new tool beyond what was in the gold standard code suite – for instance, generating a class with functions, or saving the results as a JSON.

No test written by an LLM surpassed the gold standard suite in terms of testing for something that was not tested for in the human written test suite. Responses that wrote multiple tests for data types or to check that the response is not null were ranked worse than responses with a richer variety of tests. The highest quality tests that the LLMs generated were conditional tests (for instance, checking for a value in a column if the value of another column is X), which are also included in the gold standard suite.

With regard to consistency, it was often the case that a low consistency rating was due to one of the five responses being especially different. It was rare that the LLM just entirely misinterpreted the prompt, but compared with a human data scientist, who brings a broad context to writing tests, the LLMs rarely took into account the entire context that we would like.

In some cases, one or two results would generate sentences explaining which expectations to test for instead of code, while the other results generated commented code. It was also often the case that results would test for similar things, but one result would write the code slightly different (for instance, using an if statement or a function while testing for the same thing).

The results that scored highest in consistency were exactly identical, or extremely similar with slight variations in comments or tests.

6 Discussion

6.1 On the importance and challenges of writing validation tests

Data validation is a cornerstone of high quality data science workflows. Broadly, data validation is centered around establishing the credibility of individual data values, the coherence of data with itself, and the consistency of data with relevant external sources (Alexander 2023). Data quality—or a lack thereof—can have fundamental downstream effects on the rest of the scientific process. If the data being supplied to a statistical model is incorrect in some way (e.g., a categorical variable is stored as numeric), this will have significant implications for subsequent results and conclusions (Hynes, Sculley, and Terry 2017; Breck et al. 2019; L. Zhang et al. 2023; Gao, Xie, and Tao 2016).

One way to implement data validation in practice is automated testing, which intuits data expectations from format, column name, or other meta information and converts those expectations to code (L. Zhang et al. 2023). Automated validation promotes trustworthiness and transparency. By developing a suite of tests to validate that particular characteristics of a dataset hold true, data scientists uncover fundamental assumptions that underpin their work and equip users with confidence that those assumptions are met in practice. The value of data validation testing is not limited to particular domains—it is a fundamental component of a reliable data science workflow, regardless of the data at hand (Alexander 2023; Gao, Xie, and Tao 2016).

There are a number of challenges that accompany the development and implementation of data validation tests. These challenges are particularly important when thinking about data validation efforts across research domains and levels of statistical and data science expertise.

Implementing automated tests requires the ability to develop code. There are a number of testing-centered libraries in various programming languages. `Great Expectations`—the Python library employed by Katz and Moore (2023)—contains various data testing functions and has a data assistant functionality which programmatically explores data to develop tests based on observed characteristics (Great Expectations 2023). Similarly, `pointblank` is an R package which offers predefined automated data testing functionality (Iannone and Vargas 2023). With a focus on data validation for machine learning, Hynes, Sculley, and Terry (2017) developed a data linter tool that detects possible inconsistencies in the data and suggests appropriate transformations for a specific model type. Further, Breck et al. (2019) executed an automated validation machine learning platform at Google to address concerns about the downstream effects of messy data in model training. While various data testing software have been developed and shared, many of these have limited functionality, and the implementation of a test suite may require significant data processing work and necessarily requires programming knowledge (Alexander 2023; L. Zhang et al. 2023).

From their work on data validation for the IJF Katz and Moore (2023) conclude that data tests should not be compromised just to be compatible with predefined test functions that exist in a software package. In order to develop the most valuable suite of tests, it may be necessary to create new variables in a dataset, merge multiple datasets, or create tests which are outside the scope of existing software frameworks, for example. This presents a challenge for individuals who are not trained in or comfortable with computer programming, but wish to programmatically test their data.

In addition, the development of a comprehensive, bespoke, and accurate data test suite can rarely be done by one person alone. As advocated by Katz and Moore (2023), domain expertise is fundamental to the creation and development of high-quality data tests. For instance, without incorporating the knowledge that donation dates are only collected in three jurisdictions (British Columbia, Ontario and Federal) into their code, Katz and Moore (2023)'s test for missingness for that variable would have produced inflated failure rates. This is knowledge that was gained through collaborative efforts with IJF team members who were involved in dataset construction. For an individual working on an unfamiliar dataset, seeking out domain knowledge and gaining a thorough understanding of that data is a non-trivial task in and of itself.

The process of designing, developing, and deploying data validation tests is iterative in nature and takes a great deal of time (Katz and Moore 2023). For researchers or journalists who are eager to publish their findings, for instance, there may not be enough incentive or resources for them to do this time-consuming validation work. This challenge is only amplified if an individual was not involved in the dataset construction process, does not have access to vital domain expertise, or does not have programming experience.

While data validation is valuable for the production of transparent and trustworthy research, it comes with many challenges which hinder its accessibility. This work presents an opportunity to advocate for an alternative approach to data validation through the use of LLMs. Though there has been a great deal of work done to develop more accessible programming tools for data validation, these tools do not entirely mitigate the challenges which accompany the process, namely the need for programming knowledge, domain expertise, and a great deal of time. Our findings illustrate the potential for LLMs to rapidly produce usable data validation tests, written in code, with limited information—a promising application to support the promotion of more trustworthy, transparent data workflows across domains.

6.2 On the use of LLMs in statistics and data science

In this paper we find considerable improvements with the use of few-shot learning and being more explicit about the data that we are interested in having tests for. That said, the LLM outputs that were considered are a starting place for an experienced data scientist, rather than a replacement for the data scientist.

The contributions of this experiment and this paper are twofold. Firstly, this work adds to the emerging area of inquiry of prompt engineering for large language models. In doing so, our prompts may serve as examples—both good and bad—for others hoping to implement LLMs into their research and data workflows. Secondly, this work contributes to the academic study of LLMs for the development of data science methodologies. As these tools become more powerful and the precision in our ability to specify their outputs improves, they will become tools to reduce the time and resource burden of writing software and tests.

Our work is consistent with previous literature demonstrating that LLM performance on complex, user-defined tasks is sensitive to prompt engineering. Prompting is a brittle process in which minor alterations can lead to large fluctuations in performance (Arora et al. 2022; Zhao et al. 2021), however with OpenAI's LLMs in particular, GPT-4 (Giorgi et al. 2023) seems less susceptible to this than GPT-3.5. In general, previous research has demonstrated that when using GPT-3.5, prompts should provide context (White et al. 2023; Clavié et al. 2023), be specific (White et al. 2023; Yong et al. 2023; Shieh, n.d.), define a persona or role (White et al. 2023; Clavié et al. 2023), break complex reasoning into smaller steps (White et al. 2023; Henrickson and Meroño-Peñuela 2023), and provide example responses (Brown et al. 2020; Arora et al. 2022; Shieh, n.d.) to improve performance on a variety of tasks, including classifying job types (Clavié et al. 2023), generating images for construction defect detection (Yong et al. 2023), and generating hermeneutically valuable text (Henrickson and Meroño-Peñuela 2023).

These observations are somewhat borne out in our experiment. For instance, the more specified prompt ("Describe") and the prompt eliciting step-by-step thinking ("Simulate") both show improvements over the simplest prompt ("Name"). However, improvements made by specificity break down when example data is provided ("Example"). We hypothesize that this is due to GPT-3.5 and GPT-4's poor performance on numerical and structured data in comparison to more domain-specific LLMs such as BloombergGPT which has been trained on numeric, structured financial data and utilizes a tokenizer more suited to these data (Wu et al. 2023). We do not see significant changes in performance when two different roles are specified to the LLM, which contradicts previous literature (White et al. 2023; Clavié et al. 2023). Finally, our significant improvements moving from zero- to one-shot and marginal to no improvements from one- to few-shot align with existing literature (Giorgi et al. 2023). Some studies have demonstrated GPT-3.5 and GPT-4's difficulties in responding to queries in the desired format during zero-shot structured tasks, including named entity recognition (Hu et al. 2023) and multiple choice question (Clavié et al. 2023). Giorgi et al. (2023) thus attribute performance improvements in one- and few-shot prompts to the in-context examples' guidance as to the desired output structure. Upon inspection of the outputs from zero-, one-, and few-shot prompts, we believe that this same effect exists for our prompts.

Our work additionally contributes to the growing body of work using LLMs to automate portions of the data science workflow. LLMs have already been integrated into software tools such as GitHub's Co-Pilot ("Your AI pair programmer," n.d.) and AutoGPT ("AutoGPT," n.d.) and IDEs such as IntelliJ (Krochmalski 2104) and VSCode (Dias 2023) to produce code in a variety of programming languages. These tools will continue to assist data scientists to accelerate, resolve bugs in, document, and optimize, their code in all parts of the data science workflow. LLMs have also demonstrated potential to automate other onerous or tedious tasks in data science pipelines. These include generating synthetic text data (Chung, Kamar, and Amershi 2023; Yu et al. 2023), improving search-based software testing (Lemieux et al. 2023), generating unit tests from natural language in a variety of programming languages and contexts (Lahiri et al. 2023; Schäfer et al. 2023), generating property-based tests from API documentation (Vikram, Lemieux, and Padhye 2023), and conducting exploratory data analysis (Ma et al. 2023). As LLM prompting and fine-tuning methods improve, they will reduce technical barriers to conducting transparent, accurate, reproducible analysis, enabling data scientists to be more productive and put their energy towards analysis. higher-order reasoning, and sophisticated inference. The work described in this paper towards automating data validation using LLMs has the capacity to further this evolution.

6.3 Limitations

There are a variety of limitations of this study. The most notable is that there were only three evaluators of the LLMs responses. It is possible that some of the evaluation, especially that for decency, would have benefited from additional evaluators. The study also only considers one, reasonably specific, setting. Although it is likely this is reasonably representative of many messy real-world datasets, expanding our approach to consider a variety of settings could be beneficial.

A limitation of our work on prompt engineering (and indeed of prompt engineering generally) is that we are unable to provide fully reliable explanations for improvements in performance. Though we can make convincing speculations, these are ultimately educated guesses. Previous studies have shown that in addition to being brittle to precise semantics (Arora et al. 2022; Zhao et al. 2021), prompts show improved performance from additional text which provides no new information (Henrickson and Meroño-Peñuela 2023) such as naming the model (e.g., "You are Frederick, a helpful AI"), emulating the chatbot saying that it understands the instructions, asking it to output the "right conclusion", and providing positive feedback (Clavié et al. 2023). Despite the opacity of the emergent properties of LLMs, we may be able to develop greater understanding of the impacts of prompting choices by testing these prompts on a greater diversity of datasets from across disciplines and contexts.

We were focused on writing tests in Python within the `Great Expectations` framework. It may be that testing these prompts in a variety of contexts, and also considering potential avenues of exploration beyond `Great Expectations`, will provide evidence as to whether our prompting strategies are generalizable across datasets in different formats, contexts, and technical regimes. Prompts can be conceptualized as knowledge transfer methods analogous to software patterns which are intended to provide reusable solutions to common problems (White et al. 2023). This aligns with the goal of automated data validation to reduce the need for domain expertise and resources dedicated to data validation. Thus, in order to improve the explainability of our methods and to ensure our method is generalizable, we must validate that the prompts that are effective on this dataset show similar performance on other datasets from different contexts, in different formats, and with other LLMs.

Appendix

A Prompt details

- Four prompts:

```
- The Investigative Journalism Foundation (IJF) created and maintains a
   CSV dataset related to political donations in Canada. Each
  observation in the dataset is a donation, and the dataset has the
  following variables: "index", "amount", "donor_location", "
  donation_date", "donor_full_name", "donor_type", "political_entity
  ", "political_party", "recipient", "region", "donation_year", "
  amount_monetary", "amount_non_monetary", "electoral_event", "
  electoral_district", "added".\n\rPlease write a series of
  expectations using the Python package great_expectations for this
  dataset.
- The Investigative Journalism Foundation (IJF) created and maintains a
   CSV dataset related to political donations in Canada. Each
  observation in the dataset is a donation, and the dataset has the
  following variables: "index", "amount", "donor_location", "
  donation_date", "donor_full_name", "donor_type", "political_entity
  ", "political_party", "recipient", "region", "donation_year", "
  amount_monetary", "amount_non_monetary", "electoral_event", "
  electoral_district", "added".
    - "amount" is a monetary value that cannot be less than $0. An
      example observation is "195.46". It is, possible, but unlikely
      to be more than $1,000.00. It cannot be NA. It should be a
      numeric. The maximum donation "amount" depends on the value of
      "region" and "year". For "Federal" is 1675, for "Quebec" is 100
       since 2013 and 500 for earlier years, for "British Columbia"
      is 1309.09, for "Ontario" is 3325, and for "Alberta" is 4300.
      There is no limit for "Saskatchewan".
    - "amount" should be equal to the sum of "amount_monetary" and "
      amount_non_monetary".
    - "region" can be one of the following values: "Federal", "Quebec",
       "British Columbia", "Ontario", "Saskatchewan", "Alberta". It
      cannot be NA. It should be a factor variable.
    - "donor_full_name" is a string. It cannot be NA. It is usually a
      first and last name, but might also include a middle initial.
      It should be in title case.
    - "donation_date" should be a date in the following format: YYYY-MM
      -DD. It could be NA. The earliest donation is from 2010-01-01.
      The latest donation is from 2023-09-01.
    - "donation_year" should match the year of "donation_date" if "
      donation_date" is not NA, but it is possible that "
      donation_year" exists even if "donation_date" does not. The
      earliest year is 2010 and the latest year is 2023. This
      variable is an integer.
    - "political_party" cannot be NA. It should be a factor that is
      equal to one of: "New Democratic Party", "Liberal Party of
      Canada", "Conservative Party of Canada".
```

```
    Please write a series of expectations using the Python package
        great_expectations for this dataset.
- The Investigative Journalism Foundation (IJF) created and maintains a
     CSV dataset related to political donations in Canada. Each
    observation in the dataset is a donation, and the dataset has the
    following variables: "index", "amount", "donor_location", "
    donation_date", "donor_full_name", "donor_type", "political_entity
    ", "political_party", "recipient", "region", "donation_year", "
    amount_monetary", "amount_non_monetary", "electoral_event", "
    electoral_district", "added".
    - "amount" is a monetary value that cannot be less than $0. An
        example observation is "195.46". It is, possible, but unlikely
        to be more than $1,000.00. It cannot be NA. It should be a
        numeric. The maximum donation "amount" depends on the value of
        "region" and "year". For "Federal" is 1675, for "Quebec" is 100
         since 2013 and 500 for earlier years, for "British Columbia"
        is 1309.09, for "Ontario" is 3325, and for "Alberta" is 4300.
        There is no limit for "Saskatchewan".
    - "amount" should be equal to the sum of "amount_monetary" and "
        amount_non_monetary".
    - "region" can be one of the following values: "Federal", "Quebec",
         "British Columbia", "Ontario", "Saskatchewan", "Alberta". It
        cannot be NA. It should be a factor variable.
    - "donor_full_name" is a string. It cannot be NA. It is usually a
        first and last name, but might also include a middle initial.
        It should be in title case.
    - "donation_date" should be a date in the following format: YYYY-MM
        -DD. It could be NA. The earliest donation is from 2010-01-01.
        The latest donation is from 2023-09-01.
    - "donation_year" should match the year of "donation_date" if "
        donation_date" is not NA, but it is possible that "
        donation_year" exists even if "donation_date" does not. The
        earliest year is 2010 and the latest year is 2023. This
        variable is an integer.
    - "political_party" cannot be NA. It should be a factor that is
        equal to one of: "New Democratic Party", "Liberal Party of
        Canada", "Conservative Party of Canada".

    Please simulate an example dataset of 1000 observations. Based on
        that simulation please write a series of expectations using the
         Python package great_expectations for this dataset.
- The Investigative Journalism Foundation (IJF) created and maintains a
     CSV dataset related to political donations in Canada. Each
    observation in the dataset is a donation, and the dataset has the
    following variables: "index", "amount", "donor_location", "
    donation_date", "donor_full_name", "donor_type", "political_entity
    ", "political_party", "recipient", "region", "donation_year", "
    amount_monetary", "amount_non_monetary", "electoral_event", "
    electoral_district", "added".
    An example of a dataset is:
```

```
index,amount,donor_location,donation_date,donor_full_name,
    donor_type,political_entity,political_party,recipient,region,
    donation_year,amount_monetary,amount_non_monetary,
    electoral_event,electoral_district,added
5279105,$20.00,"Granton, N0M1V0",2014-08-15,Shelley Reynolds,
    Individual,Party,New Democratic Party,New Democratic Party,
    Federal,2014,20.0,0.0,Annual,Nan,2022-11-23
    01:00:31.771769+00:00
2187800,$200.00,,,Robert Toupin,Individual,Party,Coalition Avenir
    Québec - l'Équipe François Legault,,Quebec,2018,,,,,2023-03-17
    18:02:29.706250+00:00
3165665,$50.00,,,Geneviève Dussault,Individual,Party,Québec
    Solidaire  (Avant Fusion),,Quebec,2017,,,,,2023-03-19
    18:02:24.746621+00:00
8803473,$250.00,"Nan, Nan",,Roger Anderson,Individual,Party,Reform
    Party Of Canada,Reform Party Of Canada,Federal,1994,0.0,0.0,
    Annual,Nan,2022-11-22 02:25:34.868056+00:00
2000776,"$1,425.00","Calgary, T3H5K2",2018-10-30,Melinda Parker,
    Individual,Registered associations,Liberal Party Of Canada,
    Calgary Centre Federal Liberal Association,Federal
    ,2018,1425.0,0.0,Annual,Calgary Centre,2022-11-23
    01:00:31.771769+00:00
9321613,$75.00,,2022-06-17,Jeffrey Andrus,Individual,Party,Bc Ndp,
    Bc Ndp,British Columbia,2022,,,,,2022-12-21
    02:20:49.009276+00:00
2426288,$50.00,"Stony Plain, T7Z1L5",2018-07-24,Phillip L Poulin,
    Individual,Party,Conservative Party Of Canada,Conservative
    Party Of Canada,Federal,2018,50.0,0.0,Annual,Nan,2022-11-23
    01:00:31.771769+00:00
4428629,$100.00,"Calgary, T2Y4K1",2015-07-30,Barry Hollowell,
    Individual,Party,New Democratic Party,New Democratic Party,
    Federal,2015,100.0,0.0,Annual,Nan,2022-11-23
    01:00:31.771769+00:00
1010544,$20.00,"Langley, V1M1P2",2020-05-31,Carole Sundin,
    Individual,Party,New Democratic Party,New Democratic Party,
    Federal,2020,20.0,0.0,Annual,Nan,2022-11-23
    01:00:31.771769+00:00
4254927,$500.00,"Welshpool, E5E1Z1",2015-10-10,Melville E Young,
    Individual,Party,Conservative Party Of Canada,Conservative
    Party Of Canada,Federal,2015,500.0,0.0,Annual,Nan,2022-11-23
    01:00:31.771769+00:00
8001740,$90.00,"Deleau, R0M0L0",2004-11-15,Clarke Robson,Individual
    ,Party,New Democratic Party,New Democratic Party,Federal
    ,2004,90.0,0.0,Annual,Nan,2022-11-23 01:00:31.771769+00:00

Based on this sample please write a series of expectations using
    the Python package great_expectations for this dataset.
```

- Three learning modes: zero-, one-, and few-shot learning:

```
- The following text in quotes is an example of an expectation for this
    dataset:
  """
```

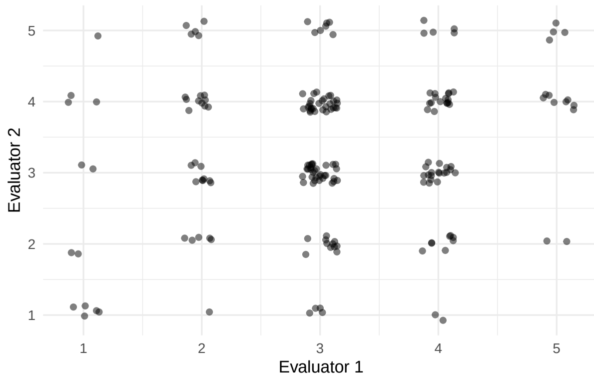
```
    # Check that there is nothing null in any column of donations
        details
    donations_mv.expect_column_values_to_not_be_null(column='
        donor_full_name')
    """
- The following text in quotes is an example of three expectations for
    this dataset:
    """
    # Check that there is nothing null in any column of donations
        details
    donations_mv.expect_column_values_to_not_be_null(column='
        donor_full_name')
    # Check that the federal donation does not exceed the maximum
    donations_mv.expect_column_values_to_be_between(
        column = 'amount',
        max_value = 1675,
        row_condition = 'region=="Federal" & donor_full_name.str.
            contains("Contributions Of")==False & donor_full_name.str.
            contains("Estate Of")==False & donor_full_name.str.contains
            ("Total Anonymous Contributions")==False & donation_year ==
             2022 & political_entity.str.contains("Leadership")==False
            ',
        condition_parser = 'pandas'
    )
    # Check that the date matches an appropriate regex format
    donations_mv.expect_column_values_to_match_regex(column = '
        donation_date',
                                                        regex = '\\d{4}-\\d
                                                            {2}-\\d{2}',
                                                        row_condition = "
                                                            donation_date.isna
                                                            ()==False",
                                                        condition_parser = '
                                                            pandas')
    """
```

- Four temperature values: 0, 0.4, 0.6, and 1.
- Two roles:

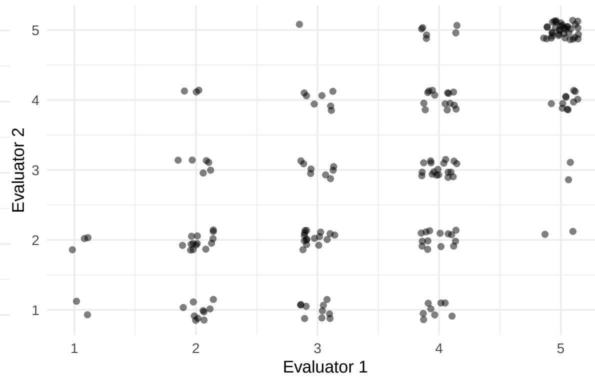
```
- You are a helpful assistant.
- You are a highly-trained, experienced, data scientist who is an
    expert at writing readable, correct, Python code.
```

B Intercoder differences

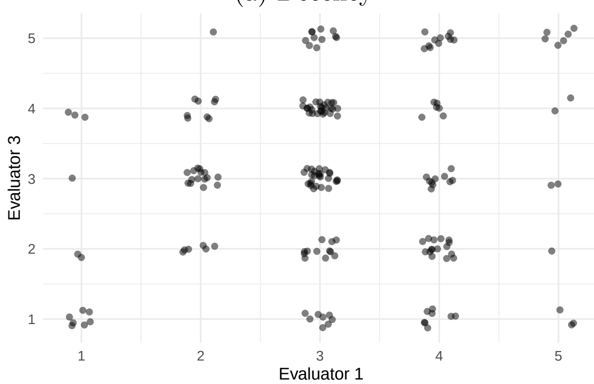
Figure 8 examines how the three evaluators rated decency and consistency. In particular, Figure 8a focuses on how they agreed on decency, and Figure 8b shows how they agreed on consistency. If all evaluators rated every response the same, then we would expect all responses to be on the 45 degree line. We do not find many absolute differences of opinion, but there nonetheless is still a wide range of differences, especially in terms of decency.



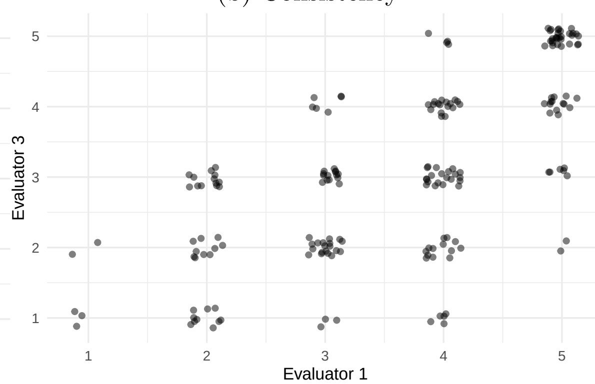
(a) Decency



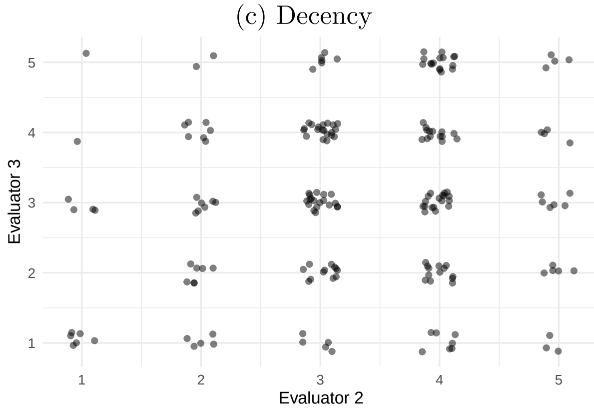
(b) Consistency



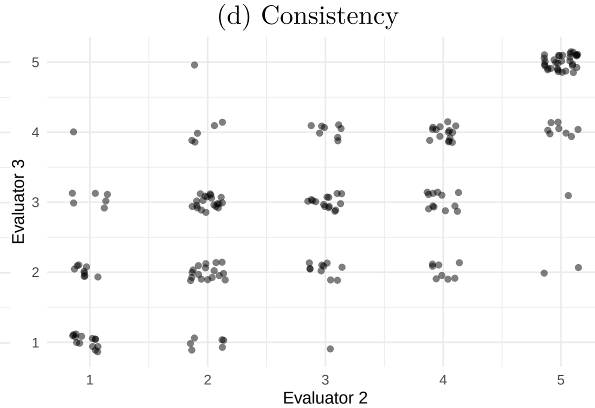
(c) Decency



(d) Consistency



(e) Decency



(f) Consistency

Figure 8: How decency and consistency are coded by each evaluator

C Model diagnostics

C.1 Consistency model

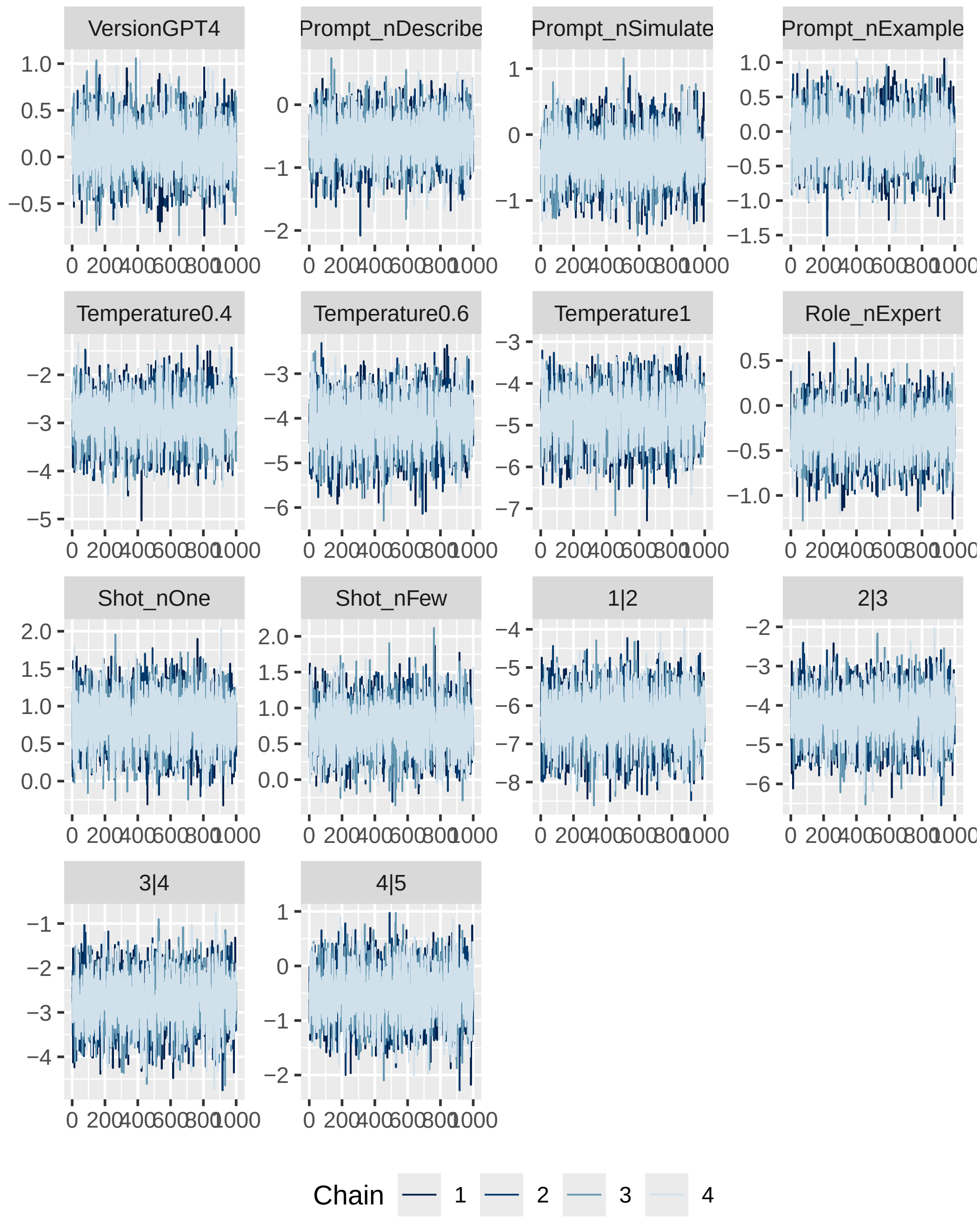


Figure 9: Consistency model trace plot

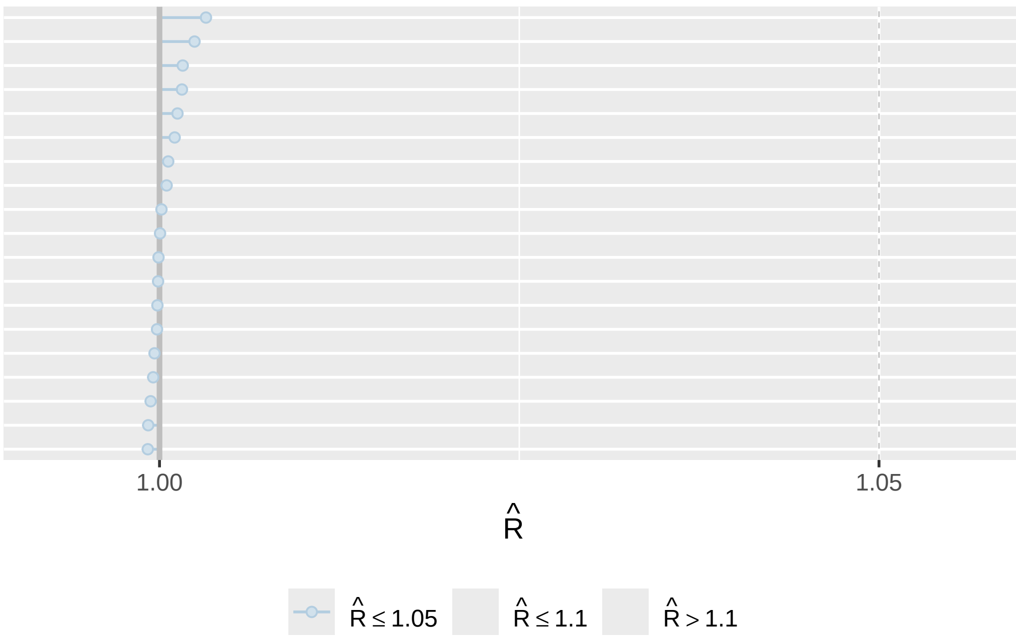


Figure 10: Consistency model rhat plot

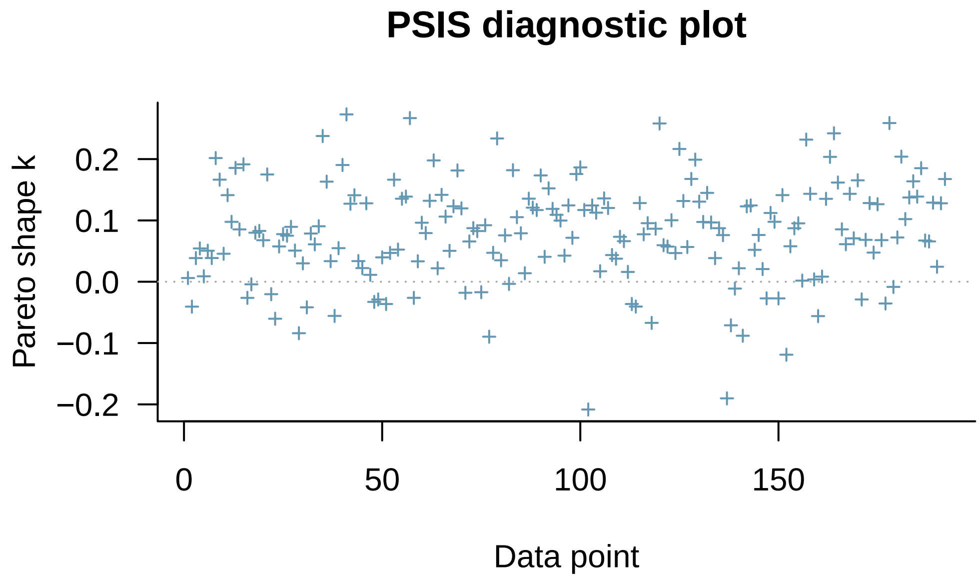


Figure 11: Consistency model loo plot

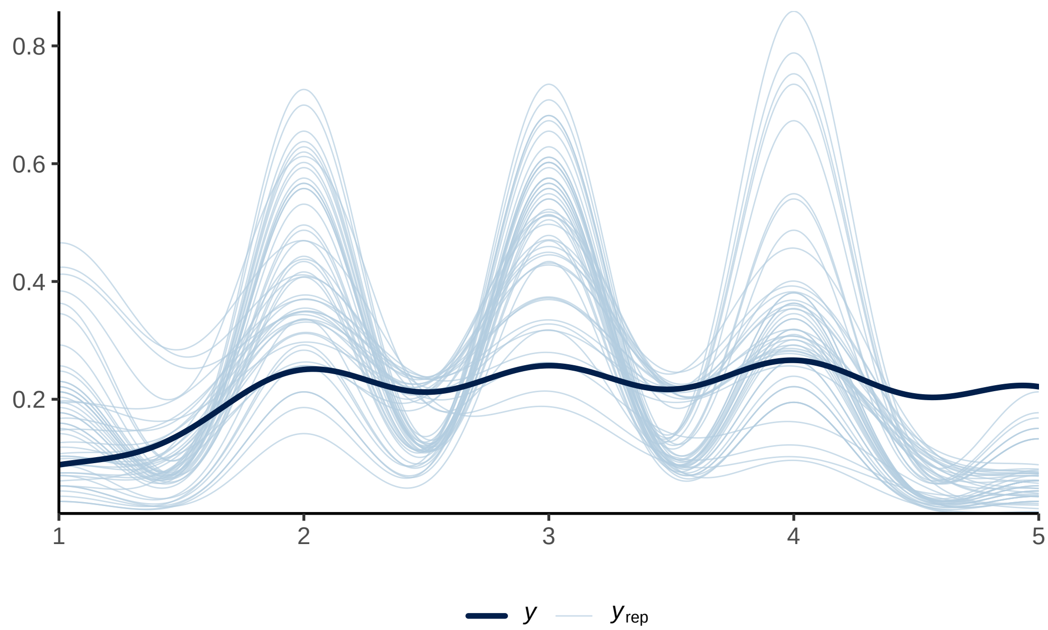


Figure 12: Consistency model posterior predictive check

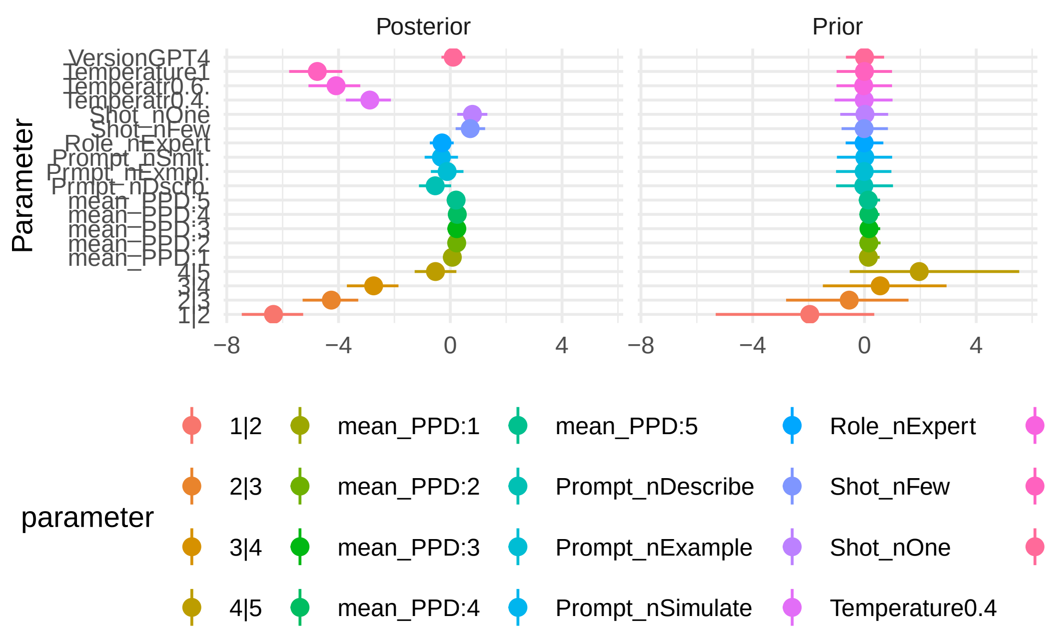


Figure 13: Consistency model posterior vs prior

C.2 Decency model

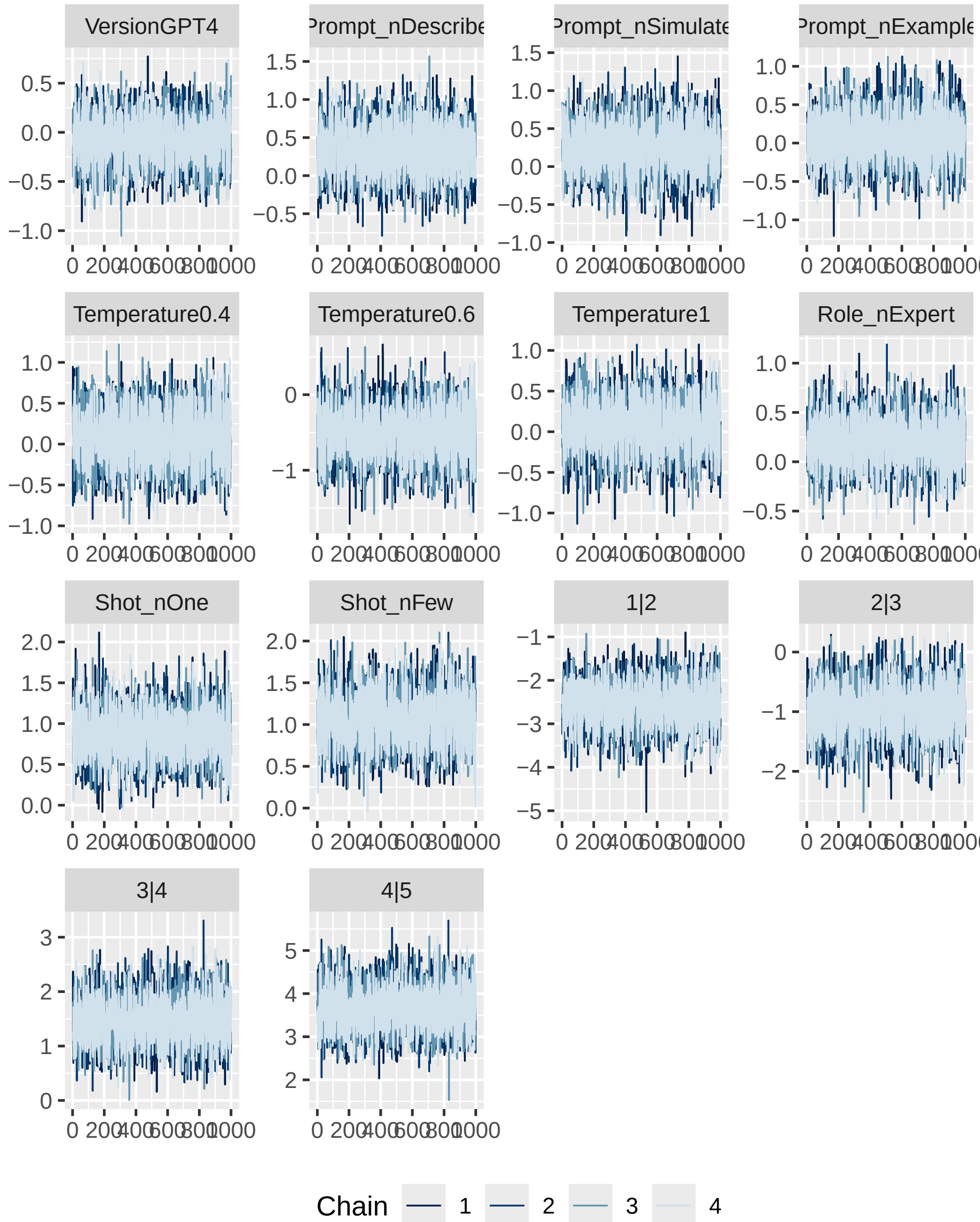


Figure 14: Decency model trace plot

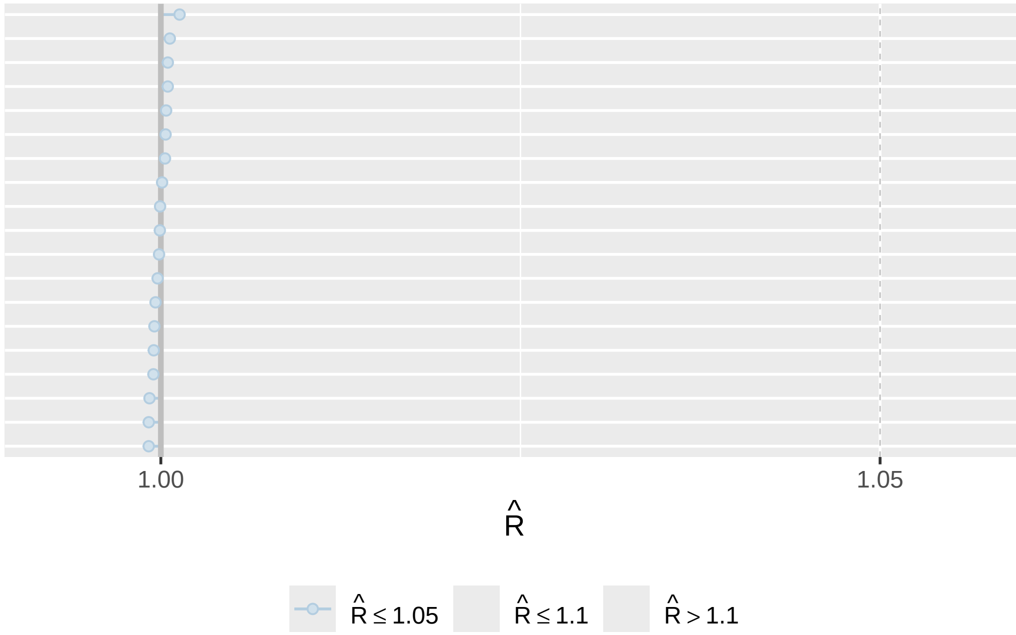


Figure 15: Decency model rhat plot

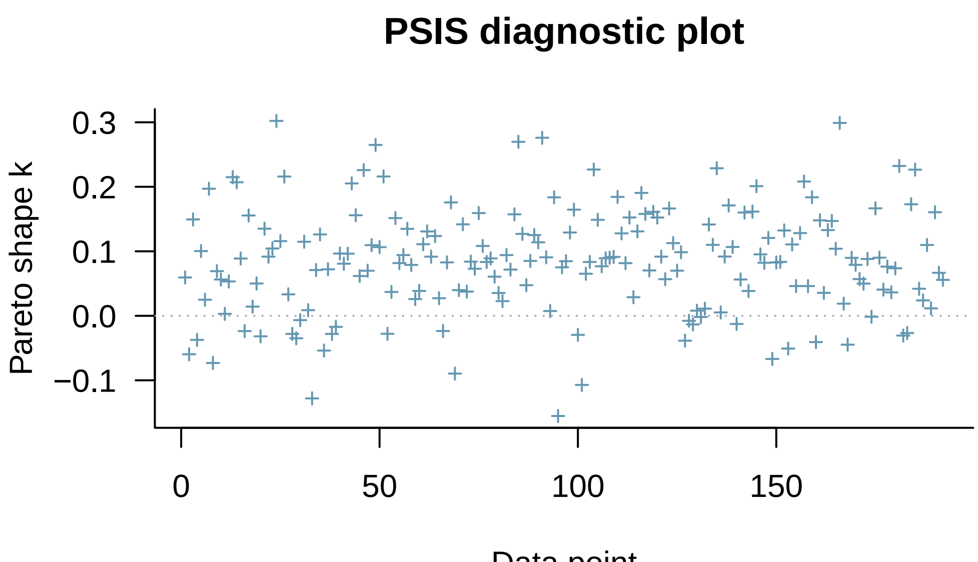


Figure 16: Decency model loo plot

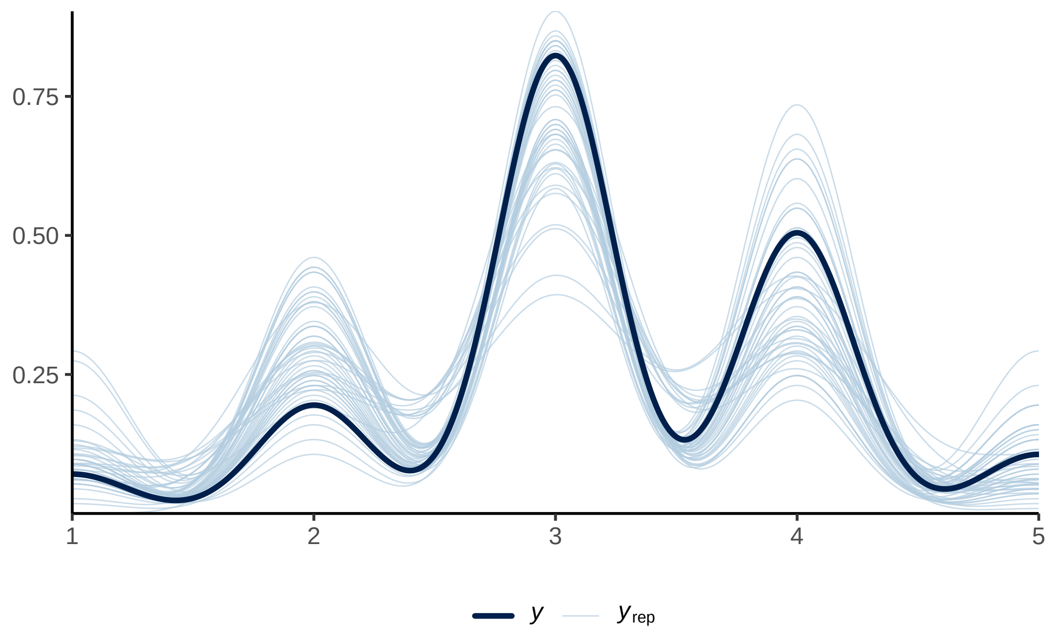


Figure 17: Decency model posterior predictive check

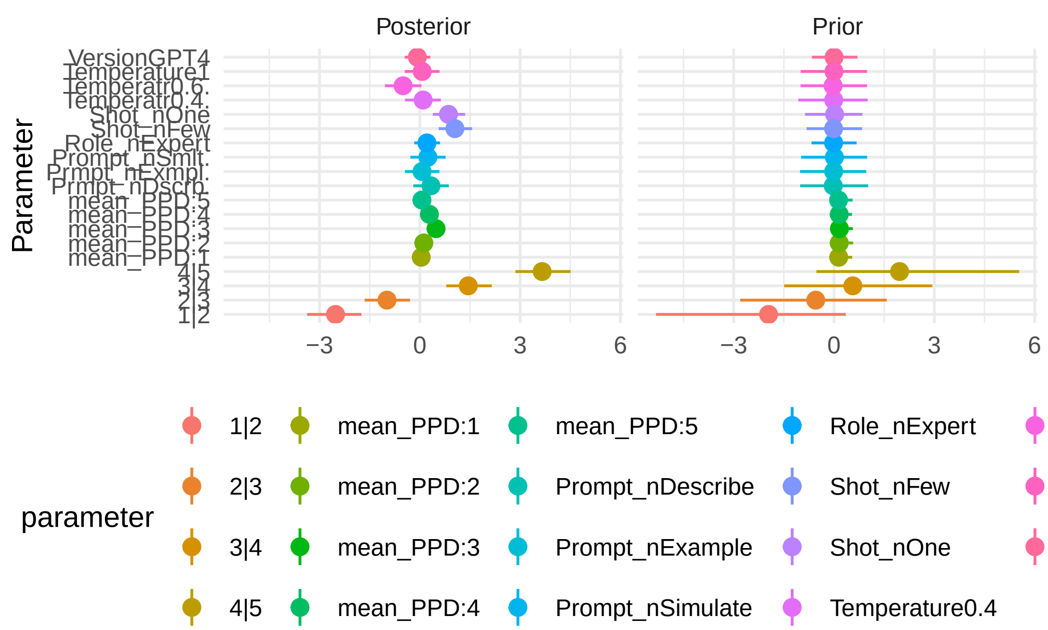


Figure 18: Decency model posterior vs prior

References

Alexander, Rohan. 2023. *Telling Stories with Data: With Applications in R*. CRC Press.

Arel-Bundock, Vincent. 2022. "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software* 103 (1): 1–23. https://doi.org/10.18637/jss.v103.i01.

Arora, Simran, Avanika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. 2022. "Ask Me Anything: A simple strategy for prompting language models." https://arxiv.org/abs/2210.02441.

"AutoGPT." n.d. https://autogpt.net/; AutoGPT.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. 2000. "A Neural Probabilistic Language Model." In *Advances in Neural Information Processing Systems*, edited by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

Breck, Eric, Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2019. "Data validation for machine learning." In *Proceedings of Machine Learning and Systems 2019, MLSys 2019*, edited by A. Talwalkar, V. Smith, and M. Zaharia, 334–47. Stanford, CA, USA: mlsys.org. https://mlsys.org/Conferences/2019/doc/2019/167.pdf.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. "Language Models are Few-Shot Learners." *CoRR* abs/2005.14165. https://arxiv.org/abs/2005.14165.

Chung, John, Ece Kamar, and Saleema Amershi. 2023. "Increasing Diversity While Maintaining Accuracy: Text Data Generation with Large Language Models and Human Interventions." In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 575–93. Toronto, Canada: Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.34.

Clavié, Benjamin, Alexandru Ciceu, Frederick Naylor, Guillaume Soulié, and Thomas Brightwell. 2023. "Large Language Models in the Workplace: A Case Study on Prompt Engineering for Job Type Classification." In *Natural Language Processing and Information Systems*, edited by Elisabeth Métais, Farid Meziane, Vijayan Sugumaran, Warren Manning, and Stephan Reiff-Marganiec, 3–17. Cham: Springer Nature Switzerland.

Dias, Chris. 2023. "Visual Studio Code and GitHub Copilot." https://code.visualstudio.com/blogs/2023/03/30/vscode-copilot; Microsoft.

Gabry, Jonah, and Ben Goodrich. 2020. *Estimating Ordinal Regression Models with rstanarm*. https://mc-stan.org/rstanarm/articles/polr.html.

Gao, Jerry, Chunli Xie, and Chuanqi Tao. 2016. "Big data validation and quality assurance–issuses, challenges, and needs." In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 433–41. IEEE.

Gelman, Andrew, Jennifer Hill, and Aki Vehtari. 2020. *Regression and Other Stories*. Cambridge University Press. https://avehtari.github.io/ROS-Examples/.

Giorgi, John, Augustin Toma, Ronald Xie, Sondra Chen, Kevin An, Grace Zheng, and Bo Wang. 2023. "WangLab at MEDIQA-Chat 2023: Clinical Note Generation from Doctor-Patient Conversations using Large Language Models." In *Proceedings of the 5th Clinical*

Natural Language Processing Workshop, 323–34. Toronto, Canada: Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.clinicalnlp-1.36.

Goodrich, Ben, Jonah Gabry, Imad Ali, and Sam Brilleman. 2023. "rstanarm: Bayesian applied regression modeling via Stan." https://mc-stan.org/rstanarm/.

Great Expectations. 2023. "Create an Expectation Suite with the Onboarding Data Assistant." 2023. https://docs.greatexpectations.io/docs/guides/expectations/data_assistants/how_to_create_an_expectation_suite_with_the_onboarding_data_assistant/.

Henrickson, Leah, and Albert Meroño-Peñuela. 2023. "Prompting Meaning: A Hermeneutic Approach to Optimising Prompt Engineering with ChatGPT." *AI & Society*, September. https://doi.org/10.1007/s00146-023-01752-8.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long short-term memory." *Neural Computation* 9 (8): 1735–80. https://deeplearning.cs.cmu.edu/F23/document/readings/LSTM.pdf.

Howard, Jeremy, and Sebastian Ruder. 2018. "Universal language model fine-tuning for text classification (ULMFiT)." https://arxiv.org/abs/1801.06146?context=cs.

Hu, Yan, Iqra Ameer, Xu Zuo, Xueqing Peng, Yujia Zhou, Zehan Li, Yiming Li, Jianfu Li, Xiaoqian Jiang, and Hua Xu. 2023. "Zero-shot Clinical Entity Recognition using ChatGPT." https://arxiv.org/abs/2303.16416.

Hynes, Nick, D. Sculley, and Michael Terry. 2017. "The Data Linter: Lightweight Automated Sanity Checking for ML Data Sets." In *NIPS MLSys Workshop*. Vol. 1. 5. http://learningsys.org/nips17/assets/papers/paper_19.pdf.

Iannone, Richard, and Mauricio Vargas. 2023. *pointblank: Data Validation and Organization of Metadata for Local and Remote Tables*. https://CRAN.R-project.org/package=pointblank.

Katz, Lindsay, and Callandra Moore. 2023. "Implementing Automated Data Validation for Canadian Political Datasets." https://doi.org/10.48550/arXiv.2309.12886.

Kim, Yoon. 2014. "Convolutional neural networks for sentence classification." https://arxiv.org/abs/1408.5882.

Krochmalski, Jaroslaw. 2104. *IntelliJ IDEA Essentials*. Packt Publishing Ltd.

Lahiri, Shuvendu K., Sarah Fakhoury, Aaditya Naik, Georgios Sakkas, Saikat Chakraborty, Madanlal Musuvathi, Piali Choudhury, et al. 2023. "Interactive Code Generation via Test-Driven User-Intent Formalization." https://arxiv.org/abs/2208.05950.

Lemieux, Caroline, Jeevana Priya Inala, Shuvendu K Lahiri, and Siddhartha Sen. 2023. "Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models." In *45th International Conference on Software Engineering*. ICSE. https://doi.org/10.1109/ICSE48619.2023.00085.

Ma, Pingchuan, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. 2023. "Demonstration of InsightPilot: An LLM-Empowered Automated Data Exploration System." https://arxiv.org/abs/2304.00477.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient estimation of word representations in vector space." https://arxiv.org/abs/1301.3781.

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, et al. 2022. "Training Language Models to Follow Instructions with Human

Feedback." *Advances in Neural Information Processing Systems* 35: 27730–44.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. "GloVe: Global vectors for word representation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://aclanthology.org/D14-1162/.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. "Deep contextualized word representations (ELMo)." In *Proceedings of NAACL-HLT 2018*. https://arxiv.org/pdf/1802.05365.pdf.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Schäfer, Max, Sarah Nadi, Aryaz Eghbali, and Frank Tip. 2023. "An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation." https://arxiv.org/abs/2302.06527.

Shieh, Jessica. n.d. "Best practices for prompt engineering with OpenAI API." https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api; OpenAI.

The IJF. 2023. "Donations Methodology." https://theijf.org/donations-methodology.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention is all you need." In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vikram, Vasudev, Caroline Lemieux, and Rohan Padhye. 2023. "Can Large Language Models Write Good Property-Based Tests?" https://arxiv.org/abs/2307.04346.

White, Jules, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT." https://arxiv.org/abs/2302.11382.

Wu, Shijie, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. "BloombergGPT: A Large Language Model for Finance." https://arxiv.org/abs/2303.17564.

Yong, Gunwoo, Kahyun Jeon, Daeyoung Gil, and Ghang Lee. 2023. "Prompt engineering for zero-shot and few-shot defect detection and classification using a visual-language pretrained model." *Computer-Aided Civil and Infrastructure Engineering* 38 (11): 1536–54. https://doi.org/https://doi.org/10.1111/mice.12954.

"Your AI pair programmer." n.d. https://github.com/features/copilot; GitHub.

Yu, Yue, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. "Large Language Model as Attributed Training Data Generator: A Tale of Diversity and Bias." https://arxiv.org/abs/2306.15895.

Zhang, Haochen, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2023. "Large Language Models as Data Preprocessors." https://arxiv.org/abs/2308.16361.

Zhang, Lei, Sean Howard, Tom Montpool, Jessica Moore, Krittika Mahajan, and Andriy Miranskyy. 2023. "Automated data validation: An industrial experience report." *The Journal of Systems & Software* 197: 111573. https://doi.org/https://doi.org/10.1016/j.jss.2022.111573.

Zhao, Tony Z., Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. "Calibrate Before

Use: Improving Few-Shot Performance of Language Models." https://arxiv.org/abs/2102.09690.