Reproducibility, Detecting Coding Errors and Robustness Analysis: Comparing the Performance of Humans, Cyborgs, and Machines with Limited Human Assistance

Abstract

This article evaluates the effectiveness of varying levels of human and artificial intelligence (AI) integration in reproducibility assessments of quantitative social science research. In study I which was conducted in 2024, we computationally reproduced quantitative results from published articles in the social sciences with 288 researchers, randomly assigned to 103 teams across three groups — human-only teams, AI-assisted teams and teams whose task was to minimally guide an AI to conduct reproducibility checks (the "AI-led" approach). In study II which was conducted in 2025, 95 researchers participated again and were randomly assigned to 34 teams. Findings reveal that when working independently, human teams matched the reproducibility success rates of teams using AI assistance, while both groups substantially outperformed AI-led approaches (with human teams achieving 52 percentage points higher success rates than AI-led teams, $p < 0.001$). Human teams were particularly effective at identifying serious problems in the analysis: on average they found significantly more major errors compared to both AI-assisted teams (0.7 more errors per team, $p = 0.017$) and AI-led teams (1.1 more errors per team, $p < 0.001$). AI-assisted teams demonstrated an advantage over more automated approaches, detecting 0.4 more major errors per team than AI-led teams ($p = 0.029$), though still significantly fewer than human-only teams. Finally, both human and AI-assisted teams significantly outperformed AI-led approaches in both proposing (25 percentage points difference, $p = 0.017$) and implementing (33 percentage points difference, $p = 0.005$) comprehensive robustness checks.

These results underscore both the strengths and current limitations of LLM-based assistance—specifically OpenAI's GPT-4/4o and 4.5/o3 as deployed in study I and II, respectively—in research reproduction. While these findings are bounded by the capabilities of the specific models used, the experimental conditions, and the time frame of study, they provide an important benchmark for what was feasible at the time. They also suggest that, in this context, these models—despite their sophistication—were not yet able to replace or significantly enhance human judgment in complex empirical workflows. While newer versions of AI tools narrowed this gap, as of now, meaningful human involvement remains essential for accurate and reliable reproducibility assessments in quantitative social science.

Replication Workflows in R and Stata

This project includes two fully independent and equivalent replication workflows: one implemented in Stata and the other in R. Both workflows use the same underlying datasets and reproduce all main tables and figures reported in the manuscript. Users may choose either language depending on their preference or software availability.

- The `code/` folder contains parallel scripts in R (with the `.R` extension) and Stata (with the `.do` extension). - All outputs generated by both workflows are identical in substance, and all results can be independently verified using either language.

While the Stata scripts use `.dta` data and the R scripts use `.rds`, both datasets are derived from the same raw source files provided in the `data/` folder.

Folder Structure

The project is organized as follows:

- **data/**: This folder contains all the raw data files necessary for analysis.

  - `AI games.xlsx`: The raw data set used for the analysis of the study.
  - `AI Games - Prompts Information.xlsx`: Raw prompt data used for study analysis.
  - `AI games.dta`: The primary data set processed used for the study's analysis in Stata format.
  - `AI games.rds`: The primary data set processed used for the study's analysis in R format.

- **code/**: Contains all scripts required for data cleaning, processing, and analysis.
  - **master**: The master script that integrates all steps of the workflow and facilitates full replication.
  - `cleaning`: Script to clean and preprocess raw data, ensuring it is ready for analysis.
  - `main`: Script to produce the primary regression tables for the study.
  - `logit poisson`: Script to produce the primary regression tables for the study using poisson and logit. There are minor differences in the SEs produced by the Stata and R versions due to the way each software handles singletons.
  - `full controls`: Script to produce the primary regression tables for the study, displaying all control variables.
  - `softwares`: Script to produce the primary regression tables for the study by software.
  - `branches`: Script to process and analyze differences across experimental branches. Only available in R.
  - `balance`: Script to generate balance tables across experimental branches. Only available in R. Only available in R.
  - `gpt skill`: Script to process and analyze differences across ChatGPT experience levels. Only available in R. Only available in R.
  - `prompts`: Script to process and analyze differences across ChatGPT usage levels. Only available in R. Only available in R.
  - `time to first`: Script to produce the density figures.
  - `reproduction rates`: Script to produce the over-time figures.

- **output/**: This folder contains the results of the analyzes, including logs and outputs.
  - `master_log.log`: A comprehensive log file capturing the execution of the main script.

Software and Tools

Required Software

1. **Stata**: Software used for data cleaning, regression analysis, and generating tables.
2. **R**: Software used for data cleaning, regression analysis, and generating tables.
3. Ran on Stata/MP 17.0 for Mac (Apple Silicon) & R 4.4.0 used in MacBook Pro M1 Max macOS Sequoia 15.1.1

R Packages

To run the R scripts, ensure the following packages are installed:

- `here`: For robust and reproducible file paths. V1.0.1
- `haven`: For reading `.dta` files. V2.5.4
- `readxl`: For importing Excel files. V1.4.5
- `rmarkdown`: For rendering R Markdown documents. V2.29
- `dplyr`: For efficient data manipulation. V1.1.4
- `tidyr`: For reshaping and tidying data. V1.3.1

- `stringr`: For string operations. V1.5.1
- `forcats`: For working with factors. V1.0.0
- `janitor`: For cleaning variable names and tabulations. V2.2.1
- `lubridate`: For working with date variables. V1.9.4
- `tibble`: For working with tibbles (modern data frames). V3.2.1
- `purrr`: For functional programming and iteration. V1.0.4
- `car`: For regression diagnostics and tools. V3.1.3
- `broom`: For tidying model outputs. V1.0.8
- `fixest`: For running regression models with fixed effects. V0.12.1
- `margins`: For calculating marginal effects. V0.3.28
- `sandwich`: For robust standard errors. V3.1.1
- `lmtest`: For hypothesis testing in linear models. V0.9.40
- `multcomp`: For multiple comparisons in regression models. V1.4.28
- `kableExtra`: For formatting LaTeX and HTML tables. V1.4.0
- `ggplot2`: For creating publication-quality graphics. V3.5.2

Stata Packages

Before running the scripts, install the following Stata packages:

- `reghdfe`: For high-dimensional fixed-effects regressions. V6.12.3.
- `ftools`: For fast data manipulation. V2.49.1.
- `estout`: For exporting regression results. V3.17.
- `ppmlhdfe`: For For high-dimensional fixed-effects poisson regressions. V2.3.0
- `rsource`: To integrate R scripts within the Stata workflow. Unique version.

Instructions to Run the Stata Workflow

1. Set Up Paths

To ensure the scripts run successfully, update the paths in `master.do` to reflect your local directory structure:

```
global path "[your/local/project/path]"
cd "$path"
```

Additionally, configure the R path variable to point to your local R installation:

```
global rpath "/path/to/R"
```

2. Run the Master Script

To execute the entire workflow:

1. Open Stata.
2. Run the `master.do` file. This script integrates all components and automates the analysis workflow, including:

3. Reproduce Specific Steps

If needed, you can run specific components of the analysis separately:

- For data cleaning:

  ```
  do "code/cleaning.do"
  ```

- For main regression tables:

  ```
  do "code/main table.do"
  ```

- For R-based analysis, open R and execute:

  ```
  source("code/branches table.R")
  source("code/balance table.R")
  ```

4. Check Outputs

The results of each step will be stored in the `output/` folder. For a detailed record of the workflow, refer to `master_log.log`.

Instructions to Run the R Workflow

1. Set Up Paths

Before running the R workflow, ensure that the working directory in `master.R` reflects the path to the project folder on your local machine:

```
main_path <- [your/local/project/path]
setwd(main_path)
```

This path is used consistently throughout the script via the `here()` package.

2. Install Required Packages

Make sure all required packages are installed (see R Packages section above). You can install any missing ones with:

```
install.packages(c("here", "haven", "rmarkdown", "readxl", "dplyr",
                   "stringr", "tidyr", "forcats", "janitor", "lubridate",
                   "fixest", "purrr", "broom", "tibble", "car", "margins",
                   "sandwich", "lmtest", "multcomp", "kableExtra", "ggplot2"))
```

3. Run the Master Script

To execute the full replication workflow in R:

1. Open R or RStudio.
2. Run the file `master.R`, located in the main directory.
3. This script will:
   - Clean and prepare the raw data.
   - Generate all main regression tables (OLS, logit, Poisson).
   - Produce supplementary tables (balance tables, branch differences, GPT skill levels, prompt usage).
   - Create key figures (e.g., reproduction rates, time to first error).
   - Write a complete log of the workflow to `output/master_log.log`.

4. Reproduce Specific Steps

To reproduce specific components of the analysis, you may source the corresponding scripts individually. For example:

```
source("code/R code/cleaning.R")          # Data cleaning
source("code/R code/main.R")              # OLS regressions
source("code/R code/branches.R")          # Branch table
source("code/R code/time to first.R")     # Figures
```

All outputs are saved automatically in the `output/` folder.

Contact Information

For more details about the replication process or to address specific issues, please refer to the corresponding author information provided in the manuscript. Alternatively, consult the supplementary materials for additional insights.