Integrating AI in Stata Programming: Perspectives on Enhancement and Constraints Across Skill Levels

Ricardo Mora[1]
Dept. Economics, UC3M

—preliminary, please do not cite—

WORKSHOP ON EMPIRICAL RESEARCH IN THE AI ERA
BPLIM

[1] Ricardo Mora acknowledges the financial support from the Spanish Agencia Estatal de Investigación (Ministerio de Ciencia e Innovación) project number PID2023-146391NB-I00.

Introduction

Large Language Models, LLMs, have exceptional natural language capabilities, including in econometric programming.

However

They **often** make **hallucinations**, i.e., confident, plausible-sounding predictions/outcomes that are incorrect/nonsensical.

- **Fabrications** appear to occur because the model prioritizes user's satisfaction (coherence, fluency, goal achievement) over factual correctness.
- **Illusory expertise hallucinations** appear to occur because the model does not know it does not know the answer.

This study focuses on **ChatGPT** "current" capabilities in assisting **Stata** (17) users.

- ChatGPT is widely popular and is acknowledged to exemplify the best of LLM abilities.
- Stata is widely used for data analysis in social sciences.

The challenge

How can ChatGPT assist Stata users?

- **Automating tasks**: repetitive or routine operations.

  LLMs excel on this front.

- **Programming tasks**: interactively writing and debugging Stata code to create software implementations.

  LLMs help highly skilled users the most.

- **Troubleshooting errors:** identify and resolve multiple issues within the code simultaneously, without iterative debugging or gradual corrections.

  This presentation focuses mainly on this last point.

The experiment goals

In the context of Stata programming, what is ChatGPT troubleshooting ability?

- To provide accurate **diagnosis** tailored to the user's knowledge level.
- To propose acceptable **solutions** and handle error-free Stata script files (".do files").
- To recognize own **limitations**, i.e., to know/suspect it does not know the answer.

I design an experiment to evaluate these skills using Stata version 17 and ChatGPT models 3.5-turbo and 4o in the context of:

- **Autonomy** within cooperation.
- Cognitive **flexibility**

Experiment design

Design

- **ChatGPT API**: Python is used to interact with ChatGPT in batch mode.
    - Ensures **independent** processing of each do file (**never** ask ChatGPT to ignore previous entries in the same chat).
- **User levels:**
    - Beginner: Foundational tasks, focusing on simple data manipulation and regression analysis (OLS). ▸ Original
    - Intermediate: Efficient data handling with loops and conditional analyses; handles multiple (IV) models with organized output. ▸ Original
    - Advanced: Extends automation through custom programs and complex workflows. ▸ Original
- **Chat GPT models**:
    - Model 3.5-turbo: Released in 2022, cheaper but nonetheless fast.
    - Model 4o: Released in 2024, more accurate.
- **Openbook (closed book)** AI model is (not) provided with additional external resources.
    - In openbook mode, the API request includes a Stata log file. In closed-book mode, the request only includes the do file.

Sample

- 225 ChatGPT API requests per user level
  - 225 = 75 do files × (3.5-turbo in closed-book mode + 3.5-turbo in openbook + 4o in openbook
  - **675** (= 225 × 3 user levels) requests in total.

- According to the error type, do files include both erroneous (≈ 95%) and error-free scripts.

- Erroneous scripts:
  - ▸ Typographical errors (≈ 1/3): mistakes made during the manual entry of code, such as misspelled words or incorrect punctuation.
  - Nonexistent ▸ Commands/options (≈ 1/3): user attempts to execute commands/options that are foreign to Stata.
  - ▸ Syntax issues (≈ 1/3):
    - Command/option code does not follow Stata grammar.
    - Erroneous overall structure of the code leads to program abortion.

▸ Detailed results

ChatGPT API Requests

First Request

- Closed-book mode

  You are an expert in Stata programming. For the uploaded do file, provide a brief explanation of why Stata gives an error message and suggest a code solution to prevent the error and avoid interruption of the do file. Be concise.

- Openbook mode

  You are an expert in Stata programming. For the uploaded do file and its corresponding log file, provide a brief explanation of why Stata gives an error message and suggest a code solution to prevent the error and avoid interruption of the do file. Be concise.

Second request (only in openbook code)

You are an expert in Stata programming. You are required to provide a response in one of the following formats ONLY: ‘Yes’, ‘No’, or ‘I do not know’. Your task is to answer the question: ‘Do you know what the error is?’ based on the provided Stata do file and its corresponding log file.

Evaluation metrics

With the sample of erroneous API requests

- **Solution effectiveness:** Percentage of requests resulting in acceptable Stata solutions.
- **Hallucination incidence:** Percentage of requests using arguments that invoke erroneous Stata behavior.
- **Illusory expertise:** Percentage of requests triggering erroneous Stata solutions conditional to ChatGPT stating that it knows the answer.

With the full sample

- **Hammer effect:** The effect on solution effectiveness of being requested "(to) provide a brief explanation of why Stata gives an error message" when there is actually no error in the script.
    - Saying that there is no error in the script counts as acceptable 'solution'.

Solution effectiveness

ChatGPT performance by openbook mode and user's level

	Closed-book mode		Openbook mode	
	Failure rate	Success rate	Failure rate	Success rate
Beginner	81.3	18.7	16.0	84.0
Intermediate	81.3	18.7	26.0	74.0
Advanced	84.0	16.0	58.7	41.3
All users	82.2	17.8	33.6	66.4

Notes: Sample of API requests with erroneous code. Failure and success rates are percentages of requests where ChatGPT does not provide an aceptable solution (Failure) or otherwise). Request under 'Closed-book mode' are requests without access to the error log file.

Success rates by error type, openbook mode, model and user's level

	Typographical errors			Command errors			Syntax errors		
	Close book	Openbook		Close book	Openbook		Close book	Openbook	
	3.5-turbo	3.5-turbo	4o	3.5-turbo	3.5-turbo	4o	3.5-turbo	3.5-turbo	4o
Beginner	45.5	95.5	100.0	10.0	90.0	100.0	6.7	60.0	83.3
Intermediate	31.8	72.7	90.9	4.0	68.0	88.0	22.2	66.7	66.7
Advanced	29.2	37.5	62.5	11.8	35.3	82.4	10.7	17.9	35.7
All users	35.3	67.6	83.8	8.1	66.1	90.3	12.9	48.2	62.4

Notes: Sample of API requests with erroneous code. Success rates are percentages of the requests correctly solved by ChatGPT.

Hallucination incidence

Some hallucinations in gpt 3.5-turbo

- The ‘summarize‘ command does not accept multiple variables to be summarized simultaneously (...).
- (...) ‘egen mean_fte = mean(fte)‘ is missing the ‘by()‘ option specifying the group over which the mean should be calculated.
- The logit command requires the option ‘nolog‘ to prevent Stata from opening a new log when the command is executed within an existing log file.
- Stata does not allow generating binary variables directly from logical expressions like ‘fte>=20‘ (...).
- (T)he ‘foreach‘ loop is not properly closed with an ‘end‘ command.
- Stata cannot calculate the average of variables that have missing values (...).

Some hallucinations in gpt 4o

- (The) ‘collapse‘ command requires the dataset to be sorted by the variables specified in the ‘by‘ option (...).
- (The) ‘gettoken‘ command (...) is not capturing the expected tokens because the ‘exog‘ list only contains three variables and cannot successfully split into ‘z1‘ and ‘z‘ (...).
- In your ‘foreach‘ loop, you’re using ‘noi‘ (noisily) instead of ‘quietly‘ or without any prefix, which is not correct for a loop definition (...).
- (The) ‘generate()‘ option is not available with the ‘tabulate‘ command in Stata (...).
- (L)ocal macro ‘depvar‘ is not correctly interpolated within the ‘run_regressions‘ command inside the loop (...).

Hallucination incidence by user's level and openbook mode

	Closed-book mode		Openbook mode	
	True argument	False argument	True argument	False argument
Beginner	30.7	69.3	94.7	5.3
Intermediate	62.7	37.3	81.3	18.7
Advanced	30.7	69.3	57.3	42.7
All	41.3	58.7	77.8	22.2

Notes: Sample of API requests with erroneous code. 'True argument' columns show percentages of requests with ChatGPT responses without hallucinations (i.e., using correct arguments). 'False argument' columns show percentage of hallucinations (i.e., responses that use factually wrong arguments related to Stata's behavior).

Hallucination incidence by error type, openbook mode, model and user's level

	Typographical errors			Command errors			Syntax errors		
	Close book	Openbook		Close book	Openbook		Close book	Openbook	
	3.5-turbo	3.5-turbo	4o	3.5-turbo	3.5-turbo	4o	3.5-turbo	3.5-turbo	4o
Beginner	36.4	0.0	0.0	80.0	5.0	5.0	83.3	10.0	0.0
Intermediate	36.4	18.2	4.5	40.0	32.0	12.0	33.3	25.9	11.1
Advanced	58.3	41.7	29.2	58.8	41.2	23.5	82.1	75.0	25.0
All users	44.1	20.6	11.8	58.1	25.8	12.9	67.1	36.5	11.8

Notes: Sample of API requests with erroneous code. Cells report hallucination rates (percentages) where ChatGPT makes factually incorrect statements about Stata behavior.

Illusory expertise

Percentages of failures conditional on expertise self-report

	Do you know what the error is?		
	(Model 3.5-turbo)		(Model 4o)
	No	Yes	Yes
Beginner	16.1	37.5	6.9
Intermediate	33.3	25.0	18.9
Advanced	75.0	61.9	44.1
All users	39.9	42.1	22.9

Notes: Sample of API requests with erroneous code. The table shows failure rates (percentages) based on user level, model, and self-reported expertise. Self-reported expertise collected through an independent request. Results for Model 4o under low self-reported expertise are excluded due to small sample sizes.

Illusory expertise under difficult tasks

	Do you know what the error is?		
	(Model 3.5-turbo)		(Model 4o)
	No	Yes	Yes
Intermediate	36.8	21.4	23.1
Advanced	78.8	66.7	46.7
All users	56.3	42.3	34.0

Notes: Sample of API requests with erroneous code from Intermediate and Advanced users who commit Command and Syntax error codes. Self-reported expertise collected through an independent request. The table shows failure rates (percentages) based on user level, model, and self-reported expertise. Results for Model 4o under low self-reported expertise are excluded due to small sample sizes.

The hammer effect

To the hammer, everything looks like a nail

- Regardless of whether the request is in openbook or closed-book mode, I prime ChatGPT to assume that the do file contains one error.

- Given the small sample size, I cannot estimate within-cell probabilities, but I can estimate a binary mode to evaluate the marginal effect of sending a conflicting instruction to ChatGPT.

- If ChatGPT has autonomous reasoning, the fact that there actually was not error in the do file should not affect the probability of appropriate answer.
  - The right answer when there is no error is something alike to "The do file can run entirely without any interruption."
  - Of course, a script that runs without interruption is not guaranteed to render the desired results. Hence, it is a ligitimate concern to offer ways to debug the script, even though it were not to stop.

Appropriate answer. Probit ML estimates.

	Full sample			Model 4o	
	Unconditional	Conditional 1	Conditional 2	Unconditional	Conditional 3
No error	-1.012***	-1.292***	-1.376***	-0.746*	-0.784*
	(0.277)	(0.317)	(0.342)	(0.407)	(0.419)
Openbook		1.150***	1.190***		
		(0.133)	(0.143)		
gpt 4o		0.578***	0.674***		
		(0.129)	(0.181)		
Command/Syntax		-0.588***	-0.585***		-0.951***
		(0.117)	(0.118)		(0.239)
Knows answer			-0.141		
			(0.186)		
N. obs.	675	675	675	225	225
AME(No error)	-0.397	-0.390	-0.415	-0.228	-0.222
	(0.105)	(0.092)	(0.099)	(0.122)	(0.116)
	[0.000]	[0.000]	[0.000]	[0.062]	[0.055]

Notes: *** $p<0.01$, ** $p<0.05$, * $p<0.1$. Standard errors in parenthesis and p-values in square brackets. Probit Maximum Likelihood estimates. Full sample includes all 625 requestes. Output variable is a dummy binary for appropriate answer. Variable 'No error' is a dummy variable that takes value 1 if the request involves a script that has no errorrs.AME(No error) is the estimated Average Marginal Effect of 'No error'. 'No error' predicts failure perfectly in the requests to gpt 3.5-turbo.

Conclusions

Conclusions

- **Solution effectiveness:** Improvements from 3.5-turbo to 4o
  - Still, large failure rates in complex do files ($\approx 50\%$) even under open book.
- Openbook model drastically reduces **Hallucination incidence**.
  - Specially true for gpt 4o.
- **Illusory expertise** is still a problem in 4o, especially under advanced programming.
- **Hammer effect:** Feeding ChatGPT with strict guidelines that challenge the openbook facts negatively influences solution effectiveness.

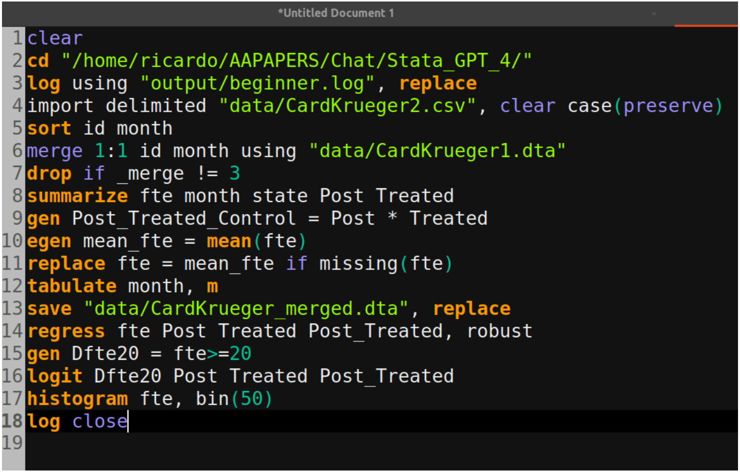
Number of observations by user level and error type

	None	Typographical	Command	Syntax	All requests
Beginner	9	66	60	90	225
Intermediate	3	66	75	81	225
Advanced	18	72	51	84	225
All users	30	204	186	255	675

Notes: Sample of API requests. The table shows the number of API requests categorized by user proficiency level ('Beginner', 'Intermediate', 'Advanced') and error type ('None': no errors, 'Typographical': minor errors in text or formatting, 'Command': usage of non-existent commands or options, and 'Syntax': structural programming errors).

▸ back

The Beginner user do file

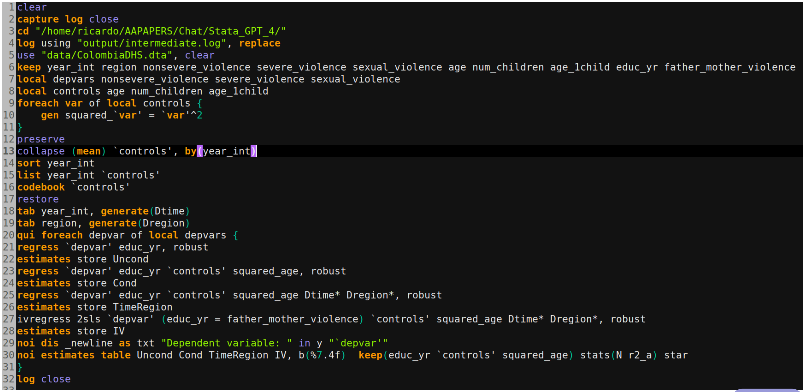


```
clear
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/beginner.log", replace
import delimited "data/CardKrueger2.csv", clear case(preserve)
sort id month
merge 1:1 id month using "data/CardKrueger1.dta"
drop if _merge != 3
summarize fte month state Post Treated
gen Post_Treated_Control = Post * Treated
egen mean_fte = mean(fte)
replace fte = mean_fte if missing(fte)
tabulate month, m
save "data/CardKrueger_merged.dta", replace
regress fte Post Treated Post_Treated, robust
gen Dfte20 = fte>=20
logit Dfte20 Post Treated Post_Treated
histogram fte, bin(50)
log close

```

▸ back

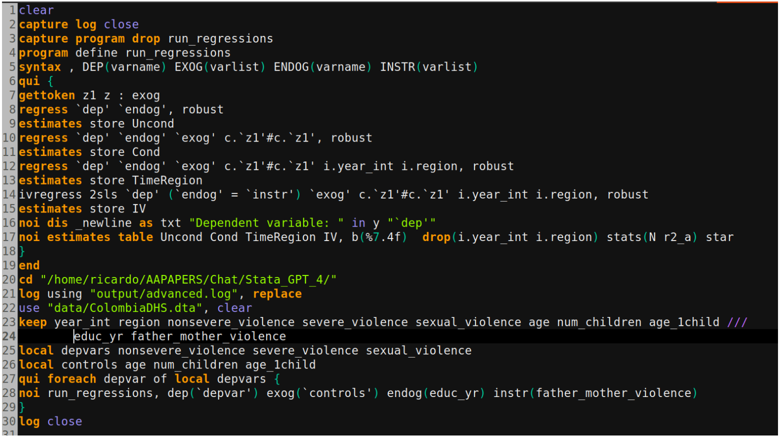
The Intermediate user do file



```
clear
capture log close
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/intermediate.log", replace
use "data/ColombiaDHS.dta", clear
keep year_int region nonsevere_violence severe_violence sexual_violence age num_children age_1child educ_yr father_mother_violence
local depvars nonsevere_violence severe_violence sexual_violence
local controls age num_children age_1child
foreach var of local controls {
    gen squared_`var' = `var'^2
}
preserve
collapse (mean) `controls', by(year_int)
sort year_int
list year_int `controls'
codebook `controls'
restore
tab year_int, generate(Dtime)
tab region, generate(Dregion)
qui foreach depvar of local depvars {
regress `depvar' educ_yr, robust
estimates store Uncond
regress `depvar' educ_yr `controls' squared_age, robust
estimates store Cond
regress `depvar' educ_yr `controls' squared_age Dtime* Dregion*, robust
estimates store TimeRegion
ivregress 2sls `depvar' (educ_yr = father_mother_violence) `controls' squared_age Dtime* Dregion*, robust
estimates store IV
noi dis _newline as txt "Dependent variable: " in y "`depvar'"
noi estimates table Uncond Cond TimeRegion IV, b(%7.4f)  keep(educ_yr `controls' squared_age) stats(N r2_a) star
}
log close
```

▸ back

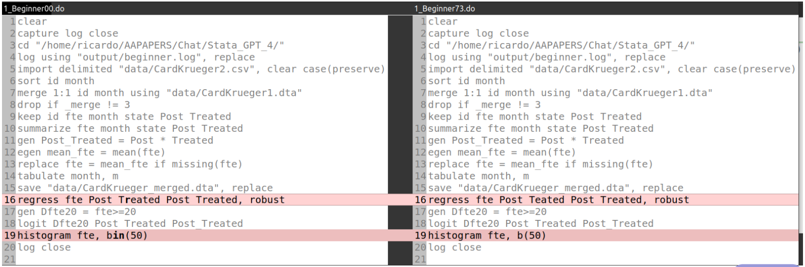
The Advanced user do file



```
clear
capture log close
capture program drop run_regressions
program define run_regressions
syntax , DEP(varname) EXOG(varlist) ENDOG(varname) INSTR(varlist)
qui {
gettoken z1 z : exog
regress `dep' `endog', robust
estimates store Uncond
regress `dep' `endog' `exog' c.`z1'#c.`z1', robust
estimates store Cond
regress `dep' `endog' `exog' c.`z1'#c.`z1' i.year_int i.region, robust
estimates store TimeRegion
ivregress 2sls `dep' (`endog' = `instr') `exog' c.`z1'#c.`z1' i.year_int i.region, robust
estimates store IV
noi dis _newline as txt "Dependent variable: " in y "`dep'"
noi estimates table Uncond Cond TimeRegion IV, b(%7.4f)  drop(i.year_int i.region) stats(N r2_a) star
}
end
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/advanced.log", replace
use "data/ColombiaDHS.dta", clear
keep year_int region nonsevere_violence severe_violence sexual_violence age num_children age_1child ///
        educ_yr father_mother_violence
local depvars nonsevere_violence severe_violence sexual_violence
local controls age num_children age_1child
qui foreach depvar of local depvars {
noi run_regressions, dep(`depvar') exog(`controls') endog(educ_yr) instr(father_mother_violence)
}
log close
```

▸ back

Typographical errors



1_Beginner00.do

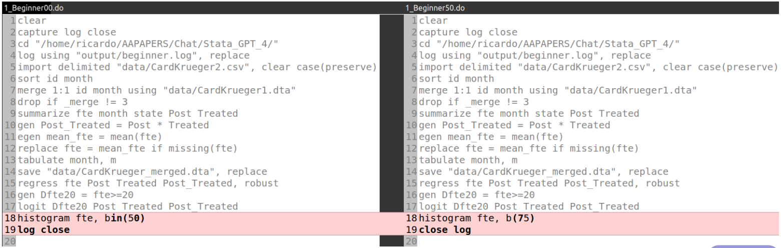
```
clear
capture log close
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/beginner.log", replace
import delimited "data/CardKrueger2.csv", clear case(preserve)
sort id month
merge 1:1 id month using "data/CardKrueger1.dta"
drop if _merge != 3
keep id fte month state Post Treated
summarize fte month state Post Treated
gen Post_Treated = Post * Treated
egen mean_fte = mean(fte)
replace fte = mean_fte if missing(fte)
tabulate month, m
save "data/CardKrueger_merged.dta", replace
regress fte Post Treated Post Treated, robust
gen Dfte20 = fte>=20
logit Dfte20 Post Treated Post_Treated
histogram fte, bin(50)
log close
```

1_Beginner73.do

```
clear
capture log close
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/beginner.log", replace
import delimited "data/CardKrueger2.csv", clear case(preserve)
sort id month
merge 1:1 id month using "data/CardKrueger1.dta"
drop if _merge != 3
keep id fte month state Post Treated
summarize fte month state Post Treated
gen Post_Treated = Post * Treated
egen mean_fte = mean(fte)
replace fte = mean_fte if missing(fte)
tabulate month, m
save "data/CardKrueger_merged.dta", replace
regress fte Post Teated Post Treated, robust
gen Dfte20 = fte>=20
logit Dfte20 Post Treated Post_Treated
histogram fte, b(50)
log close
```

▶ back

Command/options conflicts



1_Beginner00.do

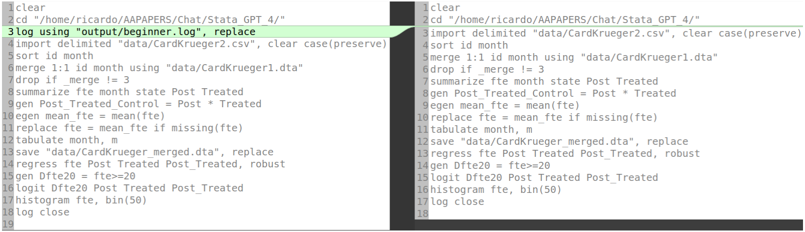
```
clear
capture log close
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/beginner.log", replace
import delimited "data/CardKrueger2.csv", clear case(preserve)
sort id month
merge 1:1 id month using "data/CardKrueger1.dta"
drop if _merge != 3
summarize fte month state Post Treated
gen Post_Treated = Post * Treated
egen mean_fte = mean(fte)
replace fte = mean_fte if missing(fte)
tabulate month, m
save "data/CardKrueger_merged.dta", replace
regress fte Post Treated Post_Treated, robust
gen Dfte20 = fte>=20
logit Dfte20 Post Treated Post_Treated
histogram fte, bin(50)
log close
```

1_Beginner50.do

```
clear
capture log close
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/beginner.log", replace
import delimited "data/CardKrueger2.csv", clear case(preserve)
sort id month
merge 1:1 id month using "data/CardKrueger1.dta"
drop if _merge != 3
summarize fte month state Post Treated
gen Post_Treated = Post * Treated
egen mean_fte = mean(fte)
replace fte = mean_fte if missing(fte)
tabulate month, m
save "data/CardKrueger_merged.dta", replace
regress fte Post Treated Post_Treated, robust
gen Dfte20 = fte>=20
logit Dfte20 Post Treated Post_Treated
histogram fte, b(75)
close log
```

back

Syntax issues



```
clear
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
log using "output/beginner.log", replace
import delimited "data/CardKrueger2.csv", clear case(preserve)
sort id month
merge 1:1 id month using "data/CardKrueger1.dta"
drop if _merge != 3
summarize fte month state Post Treated
gen Post_Treated_Control = Post * Treated
egen mean_fte = mean(fte)
replace fte = mean_fte if missing(fte)
tabulate month, m
save "data/CardKrueger_merged.dta", replace
regress fte Post Treated Post_Treated, robust
gen Dfte20 = fte>=20
logit Dfte20 Post Treated Post_Treated
histogram fte, bin(50)
log close

```

```
clear
cd "/home/ricardo/AAPAPERS/Chat/Stata_GPT_4/"
import delimited "data/CardKrueger2.csv", clear case(preserve)
sort id month
merge 1:1 id month using "data/CardKrueger1.dta"
drop if _merge != 3
summarize fte month state Post Treated
gen Post_Treated_Control = Post * Treated
egen mean_fte = mean(fte)
replace fte = mean_fte if missing(fte)
tabulate month, m
save "data/CardKrueger_merged.dta", replace
regress fte Post Treated Post_Treated, robust
gen Dfte20 = fte>=20
logit Dfte20 Post Treated Post_Treated
histogram fte, bin(50)
log close

```

▸ back

Automatic tasks

- **Problem:** reconciliation of city Chinese names between two Chinese datasets: "POI data", with 3154 places or points of interests located in 122 cities and "population data", with 299 cities. The city naming conventions were inconsistent.

- After being given the two lists , ChatGPT provided a table with a detailed reconciliation, matching city names where possible (identifying exact matches and suggesting alternatives where there were discrepancies).

- Answering the next request, ChatGPTt provided Stata code to make the two lists compatible, up to all but nine entries in the POI data. The code required minor fixes.

- A Chinese colleague quickly confirmed the reconciliation of the exact matches and we had to deal with the nine cases using traditional search tools.