REPRODUCIBILITY AND CODE

ROHAN ALEXANDER, 3 JUNE 2024, SSC

Agenda

1. Background
2. Adopting LLMs to test data
3. Adopting LLMs to improve code
4. Adopting LLMs to translate code
5. Outstanding issues

Background

“Between the idea
And the reality…
Falls the Shadow”

T.S. Eliot, The Hollow Men

Where are we?

- Virtually all applied analysis depends on code.
- We are usually required to post code alongside an article.
- Often that code is not run.
- Generally statisticians and data scientists are not trained to write code.

Where has that left us?

- Brodeur, Mikola, et al, (2024): Even excluding minor issues like missing packages or broken pathways, there are coding errors in about 25% of studies in top econ & pol sci publications since 2022.

- Xiong & Cribben (2022): Of the 93 examined papers they could only reproduce the results in 15% of fMRI-based statistics articles.

- Collins & Alexander (2022): Unable to find markers of either open data or open code for 75% of relevant pre-prints on arXiv.

- Trisovic, Lau, Pasquier & Crosas (2022): Based on more than 2,000 replication datasets with over 9,000 unique R files published from 2010 to 2020, 74% of R files failed to complete without error in the initial execution, and 56% failed even after initial code cleaning.

Can we use LLMs to improve our code?

“Yes, but”.

Data scientists have unique constraints:

1. Code typically only run a handful of times.

2. Don’t want the code to be changed too much.

3. We typically write code to analyze data so we can write a paper.

Adopting LLMs: To test data

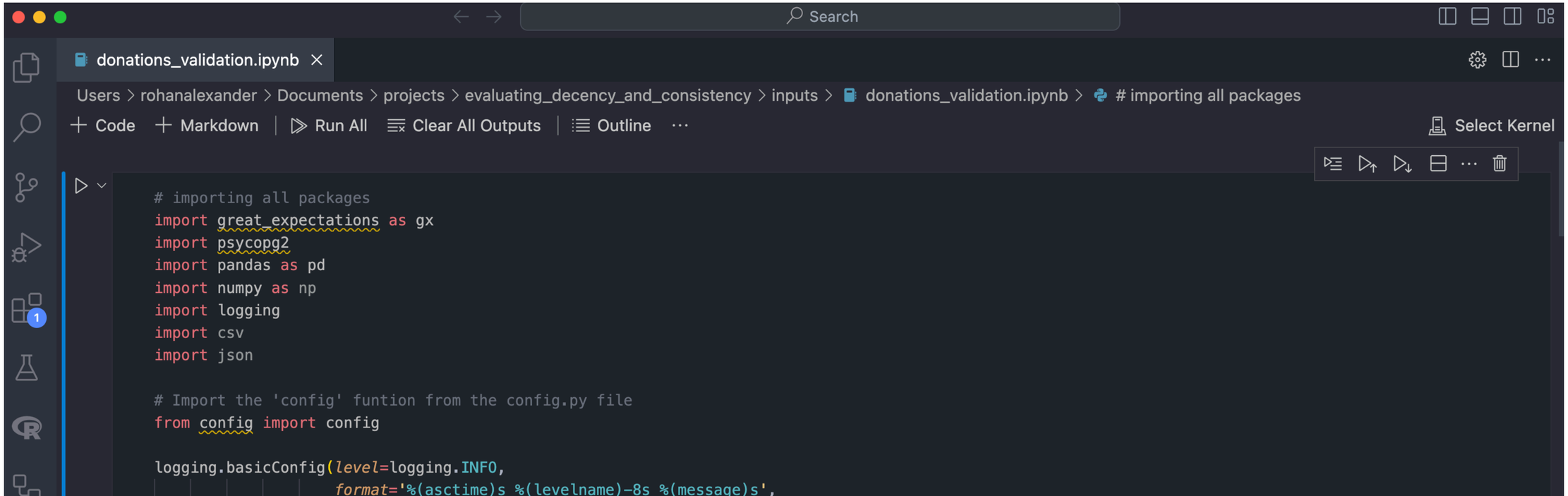
Joint work with: Lindsay Katz, Callandra Moore, Michael Wing-Cheung Wong, Zane Schwartz

Investigative Journalism Foundation

- The IJF is a nonprofit news media outlet that is centred around public interest journalism.
- One of the IJFs nine databases, and the focus of this work, is the political donations database.
- 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.
- https://theijf.org/ “Mary Moreau”

Main question

- Can LLMs develop a suite of data validation tests that is similar to the suite developed by an experienced expert data scientist who is familiar with the dataset?
- **"Yes, but".**



Set-up

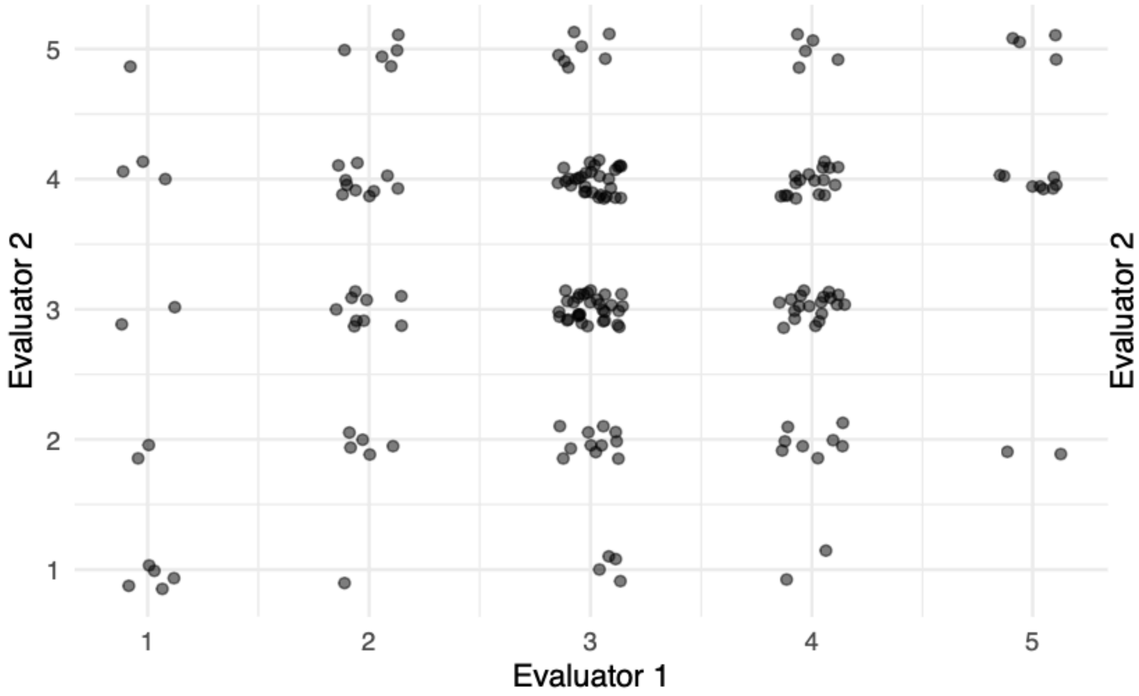
- Four prompt scenarios: 1) Asking for expectations, 2) Asking for expectations with a given context, 3) Asking for expectations after re- questing a simulation, and 4) Asking for expectations with a provided data sample.
- Three learning modes: 1) zero-shot, 2) one-shot, and 3) few-shot learning.
- Four temperature settings: 0, 0.4, 0.6, and 1.
- Two roles: 1) “helpful assistant”, 2) “expert data scientist”.
- Two models: 1) GPT-3.5 and 2) GPT-4.

Three coders evaluate: “Decency” and “Consistency

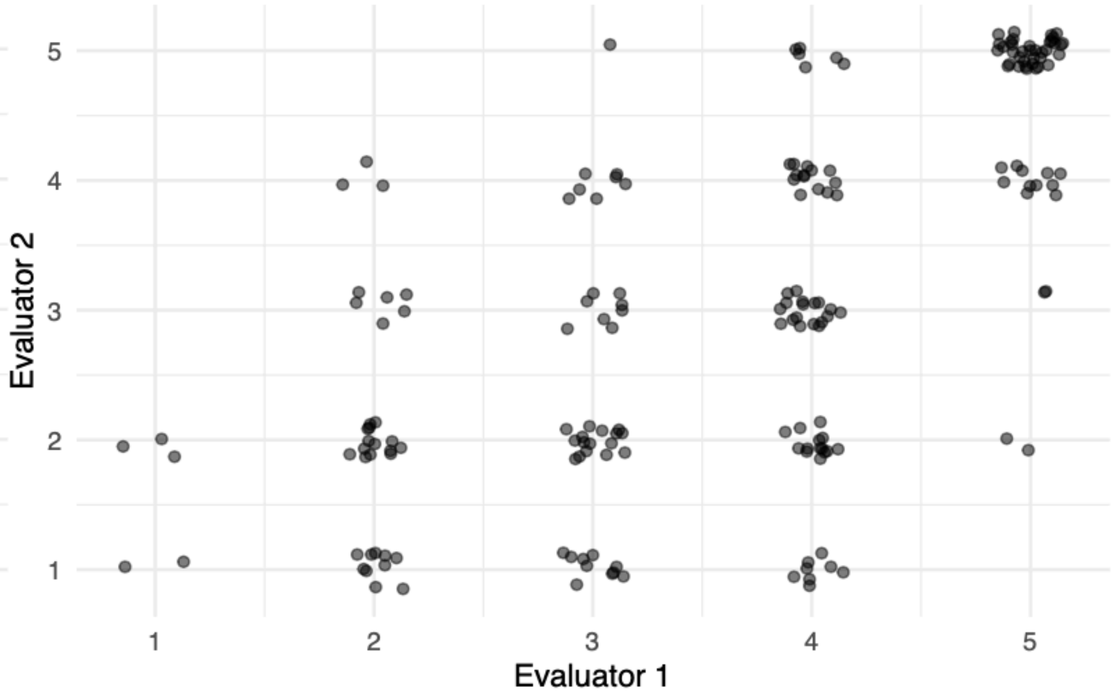
- “consistency” is a ranking 1-5 of how different each of the five responses was for that particular combination of variables.
- “decency” is a ranking of how effective the LLM validation tests were compared with the code written by the experienced data scientist who wrote the original suite of tests.

Inter-coder agreement?

Some concerns



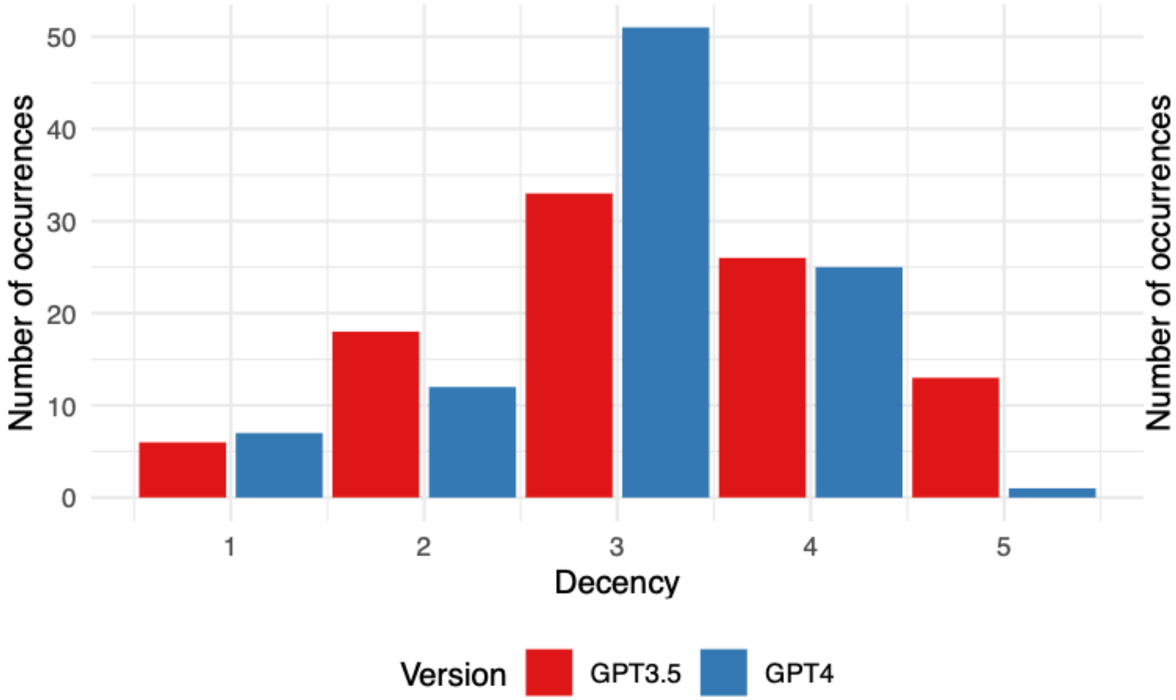
(a) Decency



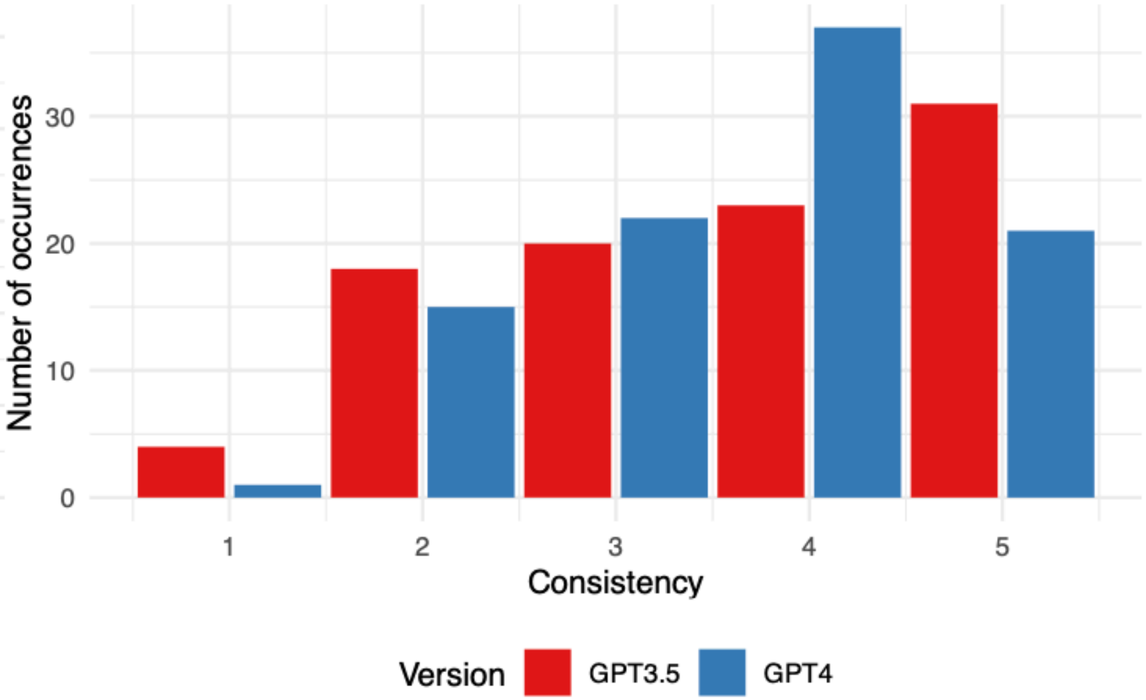
(b) Consistency

GPT-3.5 vs GPT-4

No substantial differences



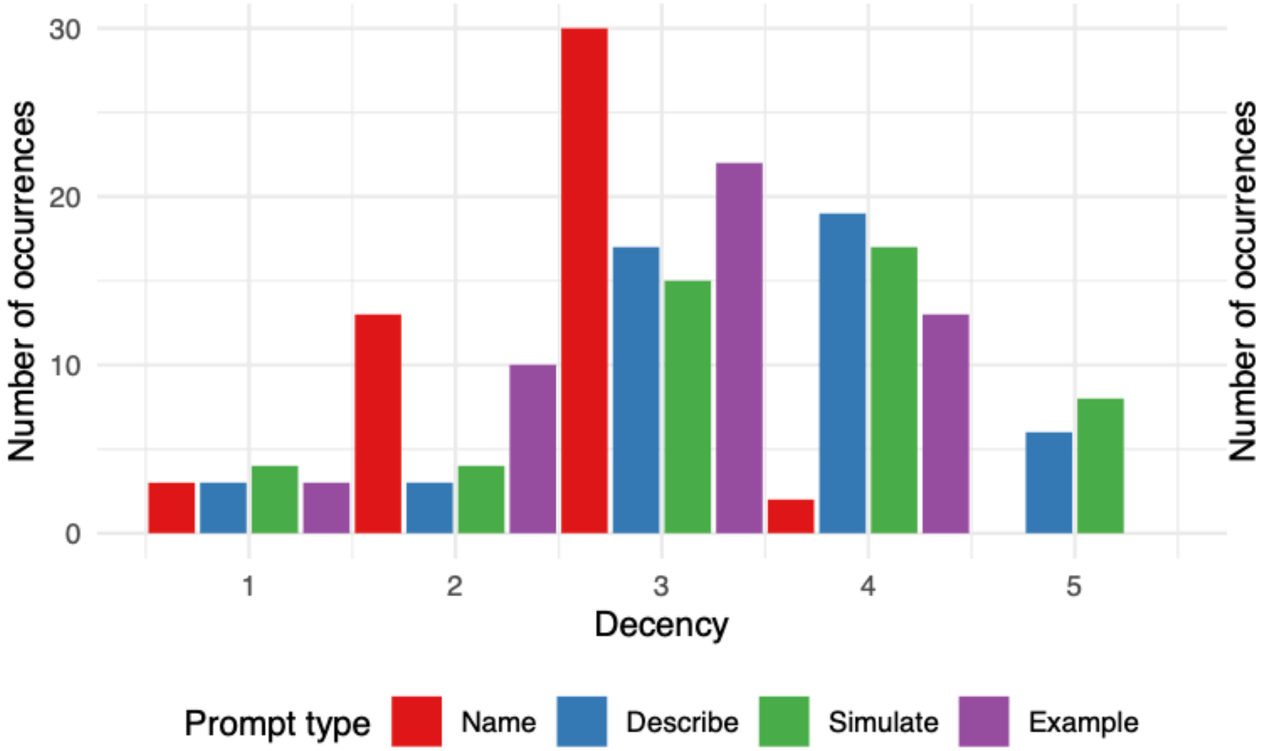
(a) Decency



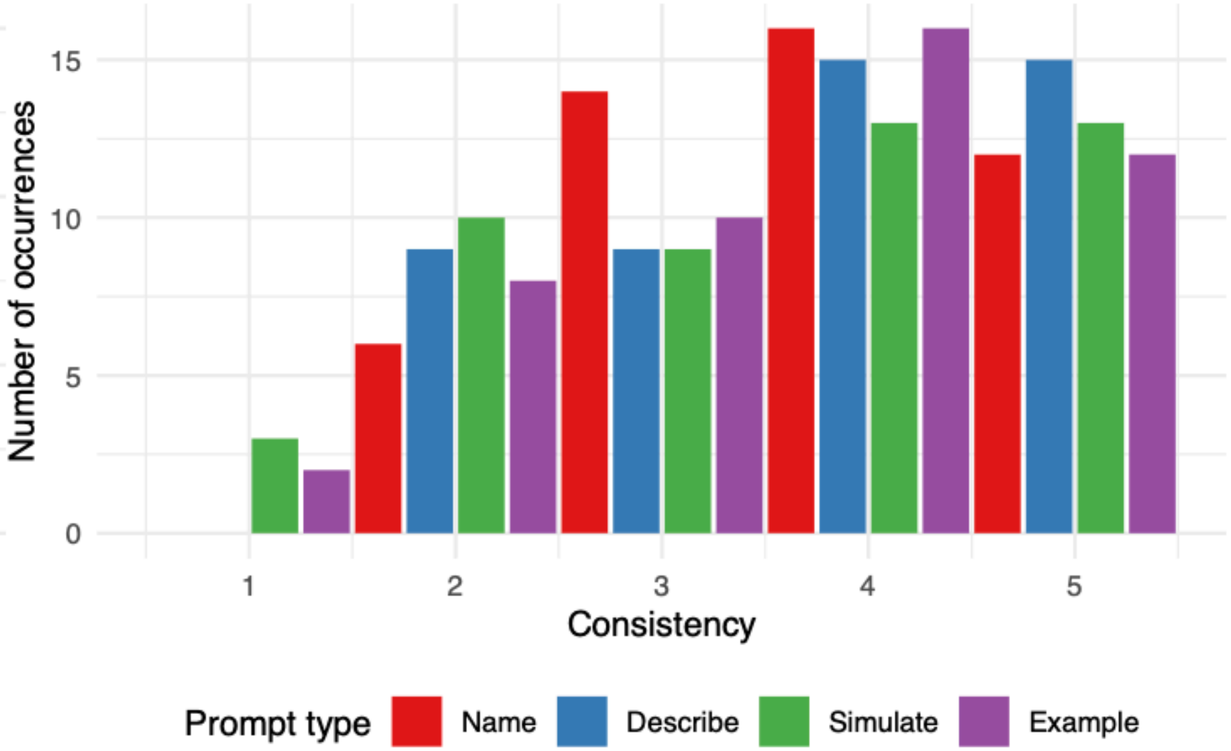
(b) Consistency

Different prompt styles

More detail is better



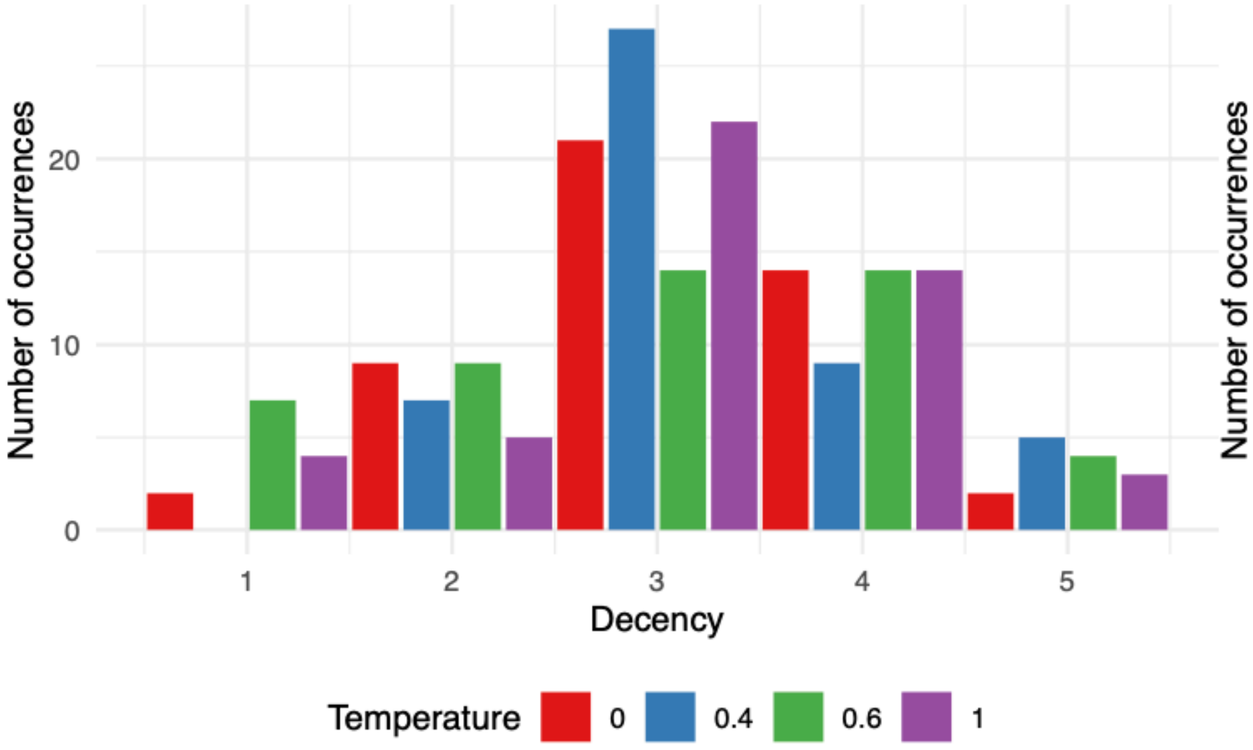
(a) Decency



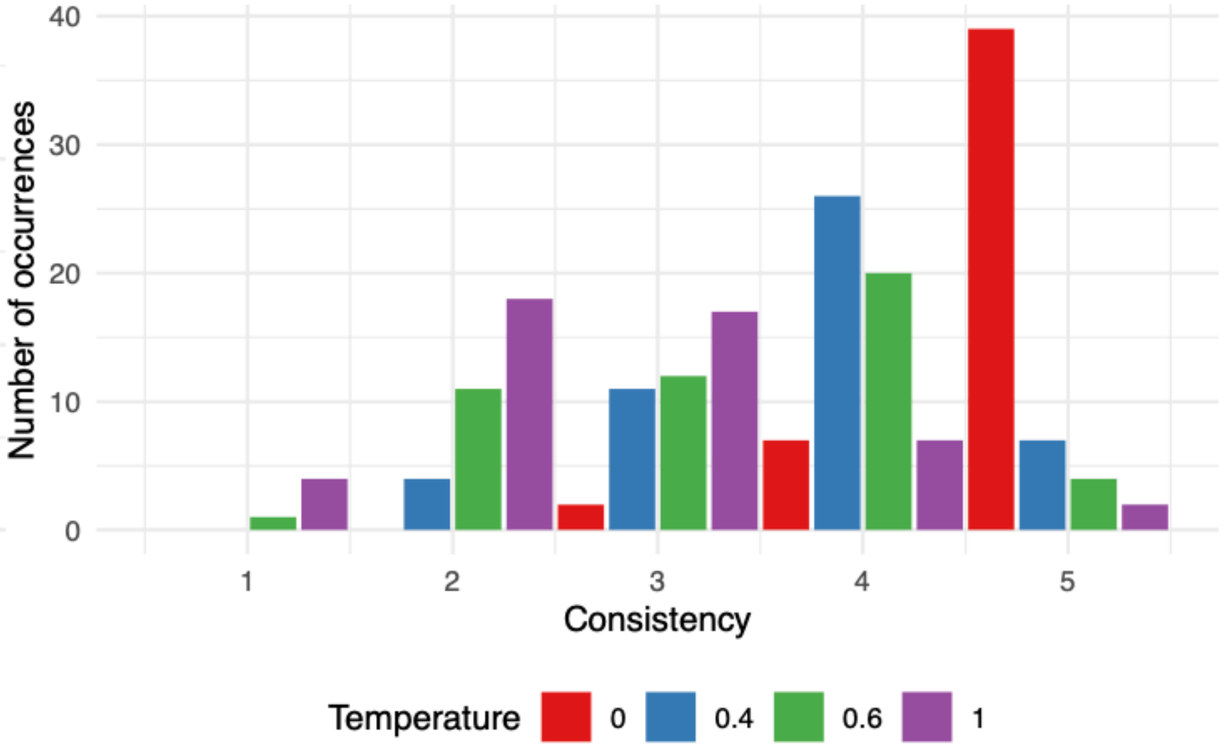
(b) Consistency

Temperature

Large effect on consistency



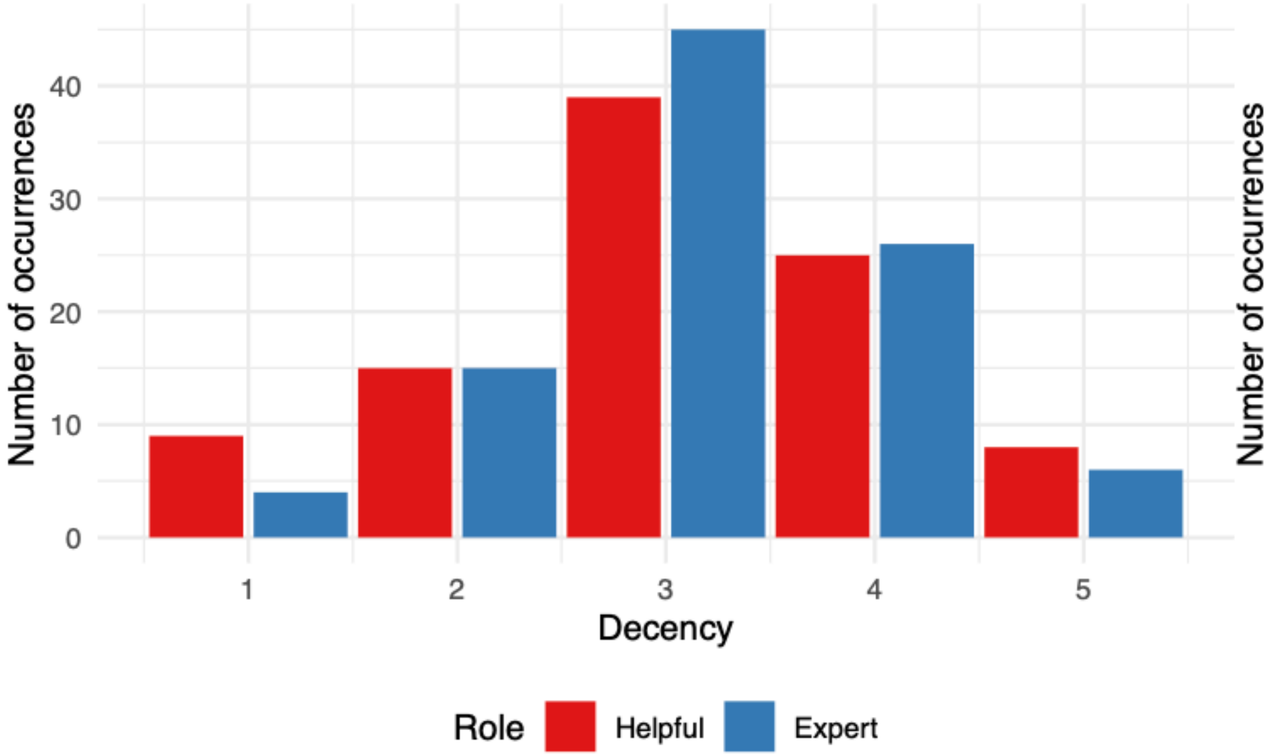
(a) Decency



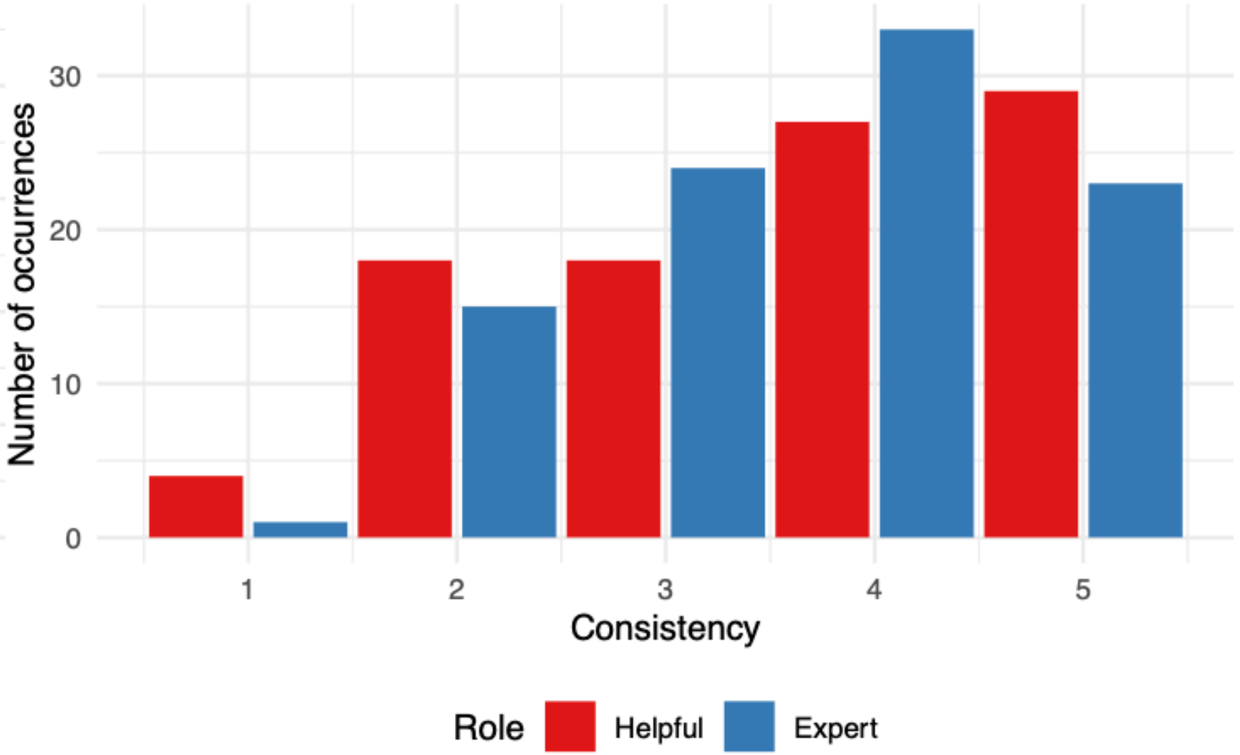
(b) Consistency

Role

Unclear effect



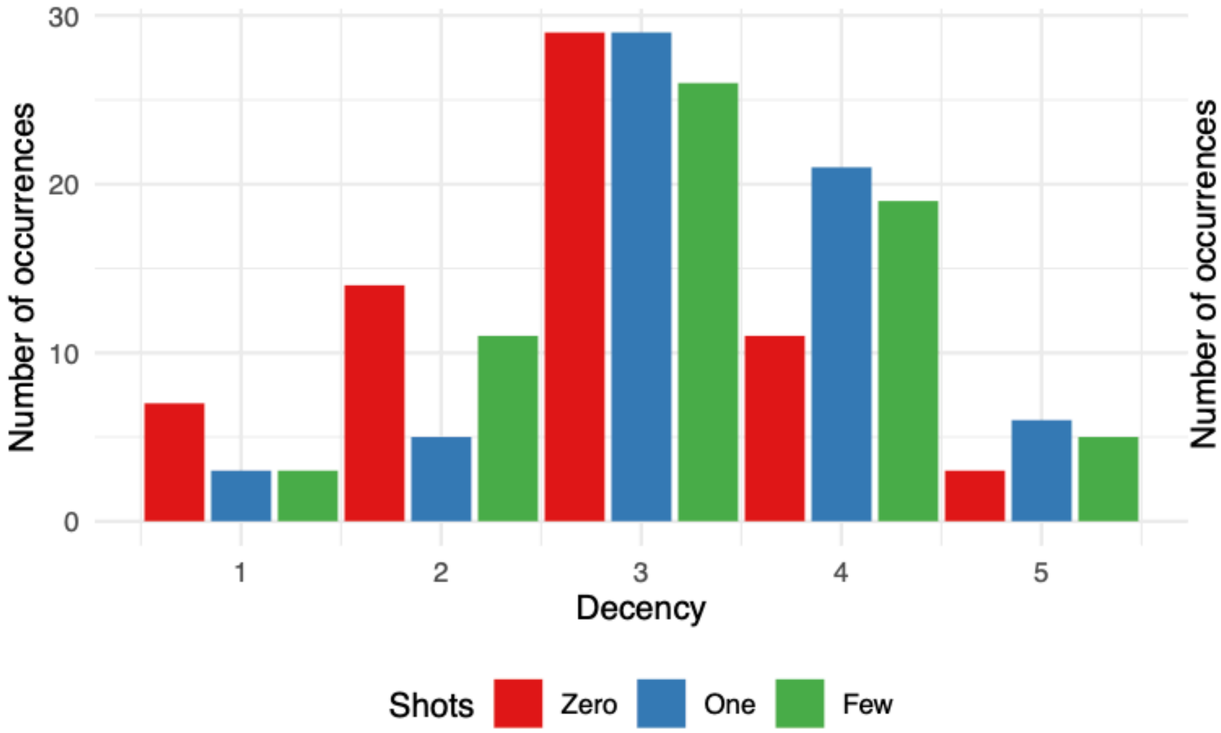
(a) Decency



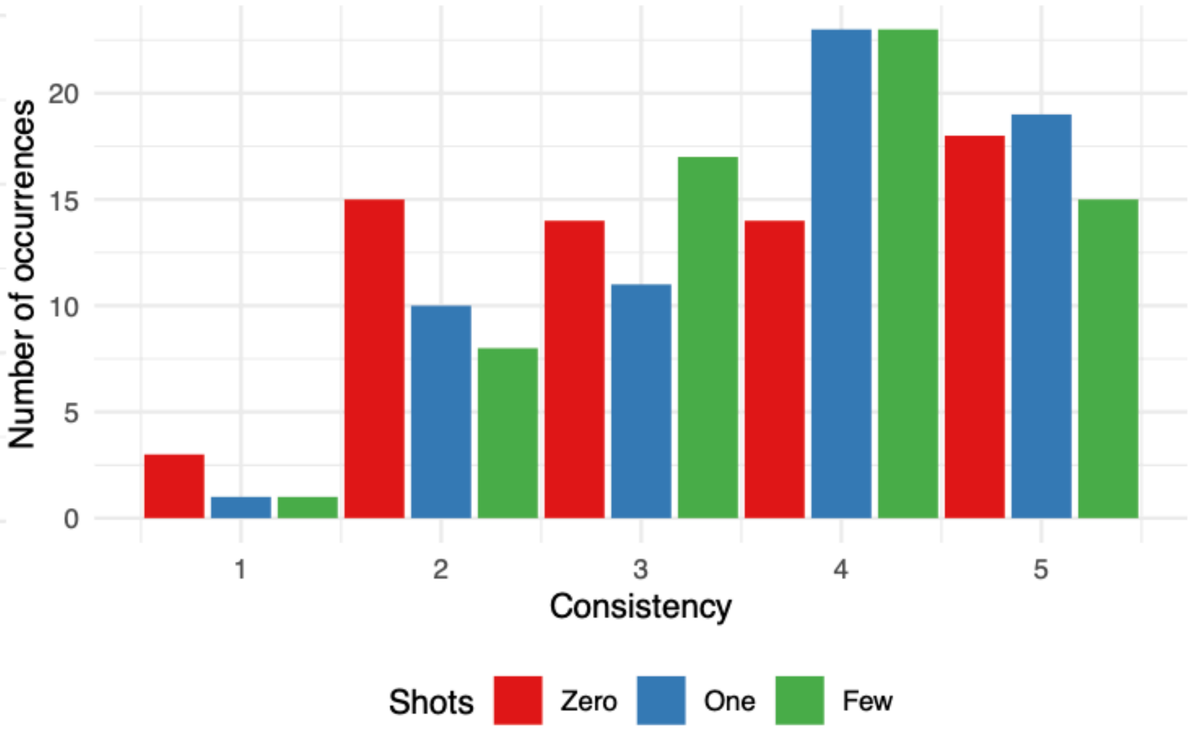
(b) Consistency

Learning mode

More examples is better



(a) Decency



(b) Consistency

Adopting LLMs: To improve our code

Joint work with: Annie Collins, Michael Chong, Zane Schwartz

Set-up

- Establish a pipeline to automatically use LLMs to improve R and Python data science code.
- Takes about 5 min to run.
- Entirely autonomously, makes a PR against a GitHub repo.

```
1    # Documentation: https://www.sbert.net/
2    # Quickstart: https://www.sbert.net/docs/quickstart.html
3    # paper: https://arxiv.org/pdf/1908.10084.pdf
4    # blog: https://blog.ml6.eu/decoding-sentence-encoders-37e63244ae00
5    # hugging face: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
6
7    from sentence_transformers import SentenceTransformer, util
8    import pandas as pd
9    from sklearn.cluster import KMeans
10   import numpy as np
11   from IPython.display import display
12
13   def get_sentence_embeddings(fName, model):
14       articles = pd.read_csv(fName)
15       headline_list = articles['title'].tolist()
16       embeddings = np.array(model.encode(headline_list))
17       print(type(embeddings[0]))
18       print("vector length", len(embeddings[0]))
40   def main():
41       model = SentenceTransformer('all-MiniLM-L6-v2')
42       # Note: initial clustering used only articles from 2021 and 2022
43       # To replicate results, "all_relevant_2021_2022.csv" should contain only
44       # a subset of headlines from "all_relevant.csv" from 2021-2022
45       embeddings, sentences = get_sentence_embeddings("development/python/example_data/all_relevant_2021_2022.csv")
46       kmeans = KMeans(n_clusters=100, random_state=0, n_init="auto")
47       kmeans.fit(embeddings)
48       centroids = kmeans.cluster_centers_
49       labels = kmeans.labels_
50       # write cluster results to file
```

```python
import os
from sentence_transformers import SentenceTransformer, util
import pandas as pd
from sklearn.cluster import KMeans
import numpy as np


def get_sentence_embeddings(file_path, model):
    '''
    Get sentence embeddings using a pre-trained model

    Arguments:
    file_path -- a string representing the file path of the input data
    model -- the Sentence Transformer model

    Returns:
    embeddings -- an array of sentence embeddings
    headline_list -- a list of headlines
    '''
    articles = pd.read_csv(file_path)
    headline_list = articles['title'].tolist()
    embeddings = np.array(model.encode(headline_list))
    print(type(embeddings[0]))


def main():
    model = SentenceTransformer('all-MiniLM-L6-v2')
    embeddings, sentences = get_sentence_embeddings(os.path.join("development", "python", "example_data", "all_
    kmeans = KMeans(n_clusters=100, random_state=0, n_init="auto")
    kmeans.fit(embeddings)
    centroids = kmeans.cluster_centers_
```

```
107
108                 articles = []
109                 time.sleep(5)
110
111     def main():
112         dates = make_dates()
113         states = [("Alabama", "AL"), ("Alaska", "AK"), ("Arizona", "AZ"), ("Arkansas", "AK"),
114                   ("California", "CA"), ("Colorado", "CO"), ("Connecticut", "CT"),
115                   ("Delaware", "DE"), ("Florida", "FL"), ("Georgia", "GA"), ("Hawaii", "HI"),
116                   ("Idaho", "ID"), ("Illinois", "IL"), ("Indiana", "IN"), ("Iowa", "IA"),
117                   ("Kansas", "KS"), ("Kentucky", "KY"), ("Louisiana", "LA"), ("Maine", "ME"),
118                   ("Maryland", "MD"), ("Massachusetts", "MA"), ("Michigan", "MI"), ("Minnesota", "MN"),
119                   ("Mississippi", "MS"), ("Missouri", "MO"), ("Montana", "MT"), ("Nebraska", "NE"),
120                   ("Nevada", "NV"), ("New%20Hampshire", "NH"), ("New%20Jersey", "NJ"),
121                   ("New%20Mexico", "NM"), ("New%20York", "NY"), ("North%20Carolina", "NC"),
122                   ("North%20Dakota", "ND"), ("Ohio", "OH"), ("Oklahoma", "OK"), ("Oregon", "OR"),
123                   ("Pennsylvania", "PA"), ("Rhode%20Island", "RI"), ("South%20Carolina", "SC"),
124                   ("South%20Dakota", "SD"), ("Tennessee", "TN"), ("Texas", "TX"), ("Utah", "UT"),
125                   ("Vermont", "VT"), ("Virginia", "VA"), ("West%20Virginia", "WV"), ("Wisconsin", "WI"),
126                   ("Wyoming", "WY"), ("GET_ALL_RELEV", "ALL_RELEV"), (None, "USA")]
127
128         for state, abrv in states:
129             get_state_results(dates, state, abrv)
130
131         # MISSING DATA FOR SOME MONTHS
132         # list of urls with errors is available in skipped-articles.txt
133         # Washington - did not collect articles due to validity concerns
134
135     main()
```

```
78    def get_state_results(dates, state, state_abrv):
103               file_code = start[0:6] + "_" + end[0:6]
104               file_name = f"development/python/example_data/gdelt_results/{state_abrv}/{state_abrv}_" + file_code + ".csv"
105               write_results(articles, file_name)
106
107               articles = []
108               time.sleep(5)
109
110   def main():
111       dates = make_dates()
112       states = [("Alabama", "AL"), ("Alaska", "AK"), ("Arizona", "AZ"), ("Arkansas", "AR"),  # Corrected the abbreviation
113                 ("California", "CA"), ("Colorado", "CO"), ("Connecticut", "CT"),
114                 ("Delaware", "DE"), ("Florida", "FL"), ("Georgia", "GA"), ("Hawaii", "HI"),
115                 ("Idaho", "ID"), ("Illinois", "IL"), ("Indiana", "IN"), ("Iowa", "IA"),
116                 ("Kansas", "KS"), ("Kentucky", "KY"), ("Louisiana", "LA"), ("Maine", "ME"),
117                 ("Maryland", "MD"), ("Massachusetts", "MA"), ("Michigan", "MI"), ("Minnesota", "MN"),
118                 ("Mississippi", "MS"), ("Missouri", "MO"), ("Montana", "MT"), ("Nebraska", "NE"),
119                 ("Nevada", "NV"), ("New%20Hampshire", "NH"), ("New%20Jersey", "NJ"),
120                 ("New%20Mexico", "NM"), ("New%20York", "NY"), ("North%20Carolina", "NC"),
121                 ("North%20Dakota", "ND"), ("Ohio", "OH"), ("Oklahoma", "OK"), ("Oregon", "OR"),
122                 ("Pennsylvania", "PA"), ("Rhode%20Island", "RI"), ("South%20Carolina", "SC"),
123                 ("South%20Dakota", "SD"), ("Tennessee", "TN"), ("Texas", "TX"), ("Utah", "UT"),
124                 ("Vermont", "VT"), ("Virginia", "VA"), ("West%20Virginia", "WV"), ("Wisconsin", "WI"),
125                 ("Wyoming", "WY"), ("GET_ALL_RELEV", "ALL_RELEV"), (None, "USA")]
126
127       for state, abrv in states:
128           get_state_results(dates, state, abrv)
129
130       # MISSING DATA FOR SOME MONTHS
131       # list of urls with errors is available in skipped-articles.txt
132       # Washington - did not collect articles due to validity concerns
133
134   main()
```

```
api_secret <-
api_key <-
authURL <-
accessURL <-
requestURL <-
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)


clean.text <- function(some_txt)
{
  some_txt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", some_txt)
  some_txt = gsub("@\\w+", "", some_txt)
  some_txt = gsub("[[:punct:]]", "", some_txt)
  some_txt = gsub("[[:digit:]]", "", some_txt)
  some_txt = gsub("http\\w+", "", some_txt)
  some_txt = gsub("[ \t]{2,}", "", some_txt)
  some_txt = gsub("^\\s+|\\s+$", "", some_txt)

  # define "tolower error handling" function
  try.tolower = function(x)
  {
    y = NA
    try_error = tryCatch(tolower(x), error=function(e) e)
    if (!inherits(try_error, "error"))
      y = tolower(x)
    return(y)
```

1. The variables api_secret, api_key, authURL, accessURL, and requestURL are declared but not assigned any values. These values need to be obtained from a specific API provider and added to the script for it to work properly.
2. The function setup_twitter_oauth() is being called with parameters access_token and access_token_secret which are not defined in the script.
3. The line `933 Tweet by Topsy` appears to be a comment or a note, but it is not properly formatted as a comment in R. Comments in R start with the `#` symbol.
4. The functions score.sentiment(), score.subjectivity(), and twListToDF() are being used in the script but their definitions are missing. These functions need to be either defined within the script or sourced from external files.
5. The file path "/Users/[redacted].csv" is specific to the author's Dropbox folder and may not work on another system. The file path should be universal or relative to the script location.

Adopting LLMs: To translate code

Joint work with: Sarah Xie, Allyson Cui, Inessa De Angelis

Monolingualism in applied work

- Quantitative social sciences are dominated by a handful of propriety languages—Stata, SAS, SPSS, Excel—e.g. as at 2019 around 60% of AEA economics papers used Stata (Vilhuber, Turrito, and Welch, 2020).
- The main issue is not the use of any particular language, but instead monolingualism.
- This allows errors unknowingly propagate and means the strengths of different languages are not taken advantage of.

(Thanks to Ash for allowing his tongue-in-cheek tweet to be used.)

Can we use LLMs to translate code?

- **“Yes, but”.** In general, “yes”: Talented undergraduates can use LLMs to quickly translate replication packages in a few hours. They quickly identify:

  1. Aspects that cannot be the actual final code.

  2. Minors errors.

- Struggles with:

  1. Language specific packages such as binscatter.

  2. Inconsistencies between different files.

  3. Differences between what is in the paper and what is in the code.

- For now, the key is the “talented undergraduates” bit! They quickly iterate and improve the results of the LLMs.

Monolingualism

Stata results:

	Illiteracy							
	Argentina, Brazil, and Paraguay		Brazil		Argentina		Paraguay	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Mission distance	0.0105***	0.0112**	0.0200***	0.0313***	0.0157**	0.0669***	0.00451	0.0138
	(0.004)	(0.005)	(0.007)	(0.010)	(0.007)	(0.022)	(0.012)	(0.027)

Python results:

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
VARIABLES	illiteracy	illiteracy	illiteracy	illiteracy	illiteracy	illiteracy	illiteracy	illiteracy
distmiss	0.0105***	0.0112**	0.0200***	0.0313***	0.0157**	0.0669***	0.00451	0.0138
	(0.00368)	(0.00495)	(0.00653)	(0.00985)	(0.00735)	(0.0220)	(0.0120)	(0.0270)
Constant	-35.33***	-53.74*	1,725	731.8	69.26*	-41.06	8.673***	-80.72*
	(11.80)	(32.50)	(1,159)	(1,299)	(36.79)	(54.63)	(0.694)	(43.89)
Observations	549	548	467	467	42	42	40	39
R-squared	0.042	0.073	0.056	0.095	0.165	0.669	0.004	0.251

Monolingualism

Typos: The dta is “Structural Transformation Brazil.dta”

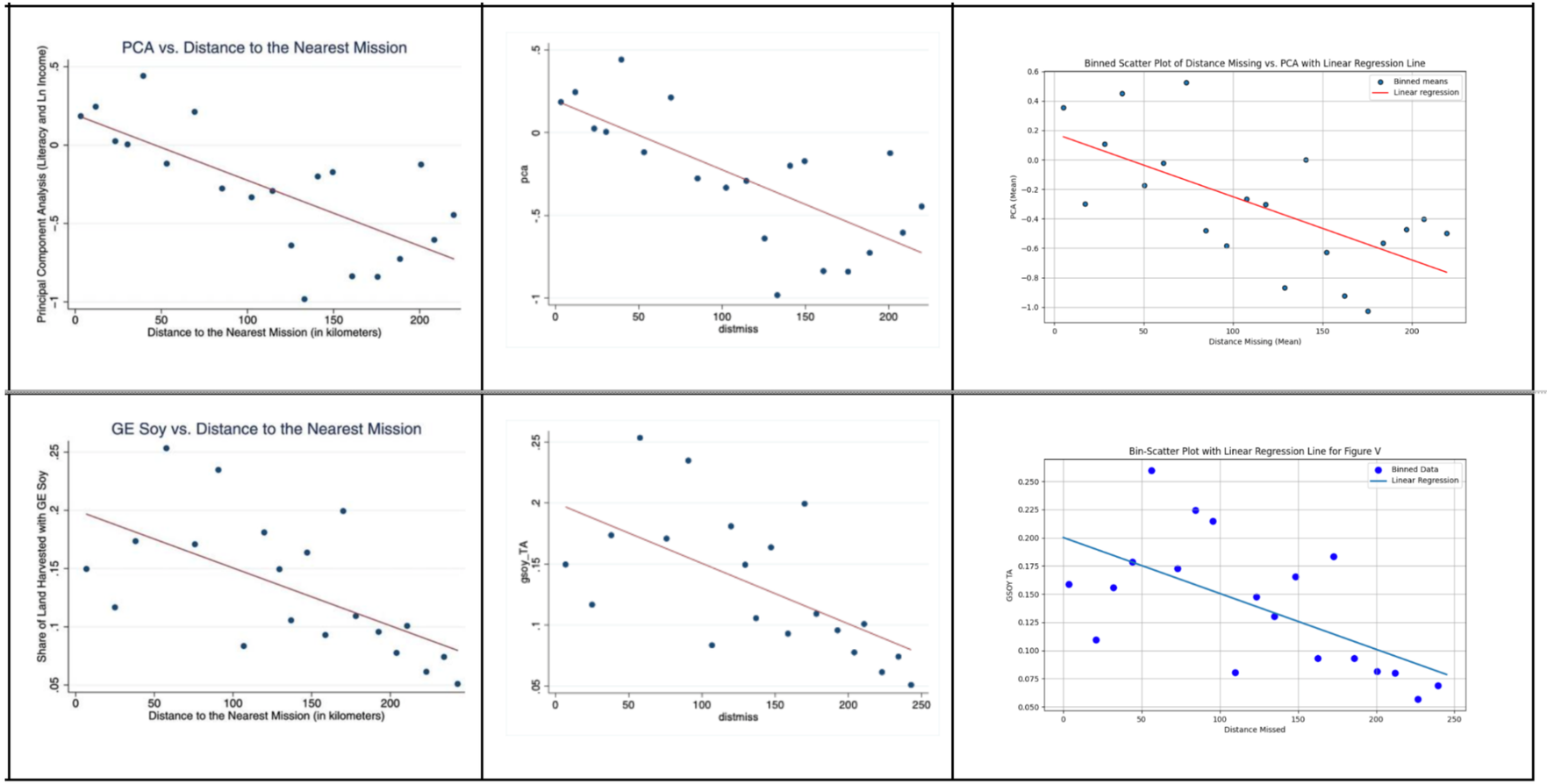
```
292   *Panel B
293   *Brazil
294   *use Brazil Structural Transformation.dta.dta
295   qui: reg agroper distmiss distfran lat1 long1 area temp alt rain rugg river coast slope meso, r
296   qui: outreg2 using TableIXB, append keep(distmiss)
297   qui: reg induper distmiss distfran lat1 long1 area temp alt rain rugg river coast slope meso, r
298   qui: outreg2 using TableIXB, append keep(distmiss)
299   qui: reg commper distmiss distfran lat1 long1 area temp alt rain rugg river coast slope meso, r
300   qui: outreg2 using TableIXB, append keep(distmiss)
301   *Conley errors calculated using Conley Standard Errors.do
302   *Within R-squared calculated manually
```

Syntax issues: Line 394 is missing command

```
388   *use Pre-Colonial Population Density.dta
389   *Data is from Maloney and Valencia Caicedo (2016)
390   *Available at: https://onlinelibrary.wiley.com/doi/full/10.1111/ecoj.12276
391   qui: iis countrynum
392   qui: xtreg popd mission temp_avg rainfall alti landlocked altisq tempsq rainsq , r fe
393   qui: outreg2 using TableXIIA, append keep(mission)
394   popd mission temp_avg rainfall alti landlocked altisq tempsq rainsq
395   *reg popd mission temp_avg rainfall alti landlocked altisq tempsq rainsq arg bra, cl (country)
396   *Within R-squared calculated manually
```

Monolingualism

Python did not adapt well with binscatter (packages unique to the language)



Some outstanding issues

- What does an effective test suite look like in data science?
- How can tests address the issue of (lack of) data sharing?
- What does a test checklist look like? (Ask Tiffany!)
- How do we develop expectations of coefficients before running the model?
- How can we integrate model tests with notions of priors?
- How can we automate the creation of model tests?
- How can we automate the creation of data tests?
- Data issues as an under-specified type system. To what extent do we need, say, an interest rate class instead of numeric.