MACS 30112

Winter 25

Final Project

Global Spatial Distribution of Open-Source Artificial Intelligence

Anqi Wei, Baihui Wang, Charlotte Li

MACS 30112

Winter 25

Final Project

Guidelines

01 Introduction

02 Data Collection

03 Data Cleaning

04 Data Visualization

05 Data Wrangling

06 Data Analysis

07 Conclusion

MACS 30112

Winter 25

Final Project

Introduction

Topic Description, Research Question & Hypothesis

Introduction

Description of Topic

Key Focus:

- Analyze global spatial patterns of open-source AI development.
- Explore socioeconomic drivers (GDP, education, scholar publication).

Significance:

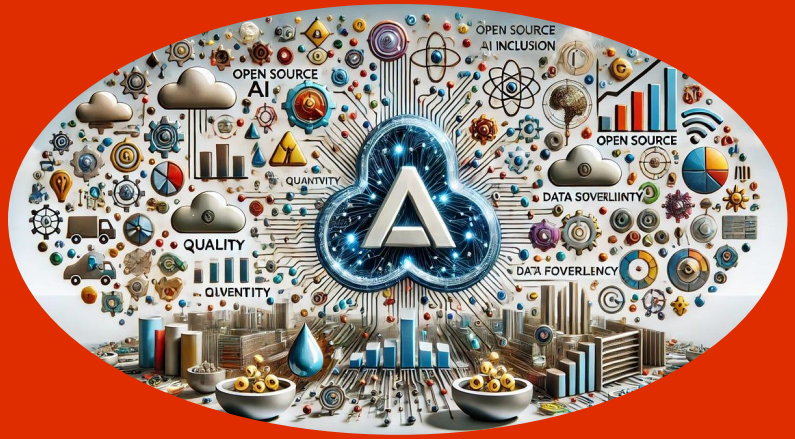
- Democratizing AI innovation vs. persistent geographic inequality.
- Policy implications for equitable technological access.

RQ1: Which cities/countries dominate AI-related open-source projects?

RQ2: How do economic development and higher education explain these patterns?

Introduction - Research Question

Hypothesis



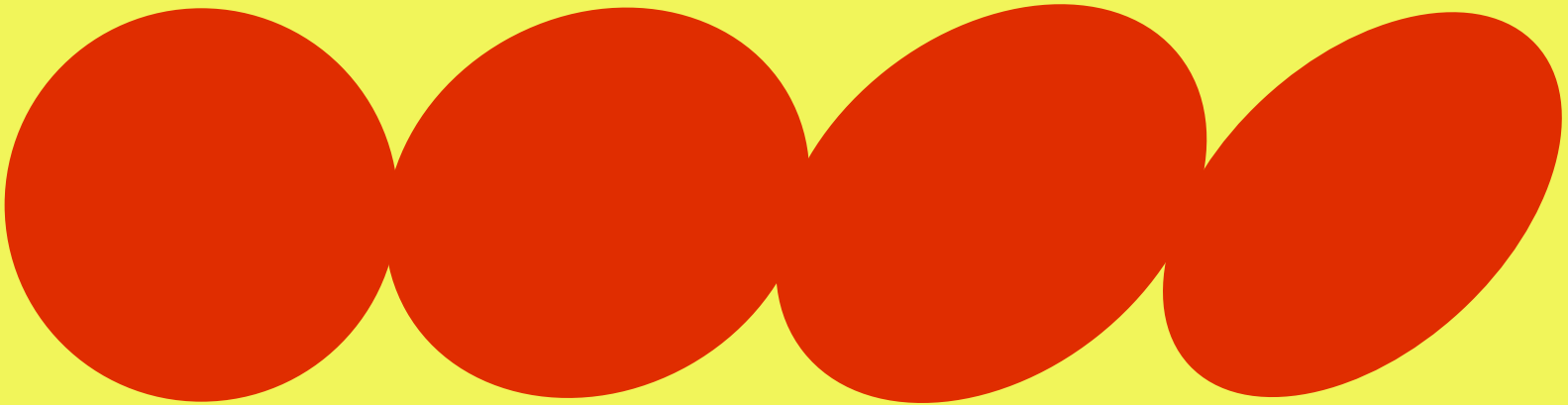
- **H1:** Higher GDP density → Higher open-source AI activity.
- **H2:** Strong AI publication presence → Reinforces open-source contributions.

MACS 30112

Winter 25

Final Project

Data Collection



Data Sources, Data Collection & Scraping

Data Sources

Source	Description
GitHub	Top 270 AI repositories (star-based ranking); 11,256 contributors' locations.
World Bank	GDP & GDP per capita (2023)
UNESCO GERD	R&D Investment as % of GDP (proxy for innovation capacity).
UN Population Data	City/country population metrics for per capita calculations.
Our World in Data	Annual AI Publications for tracking AI research output.

Data Collection

Data Collection & Scraping

GitHub Workflow:

- Scraped AI repositories using Python (pagination + token rotation to bypass API limits).
- Extracted metadata: stars, contributors, commit frequency, geolocations.

Geocoding:

- OpenStreetMap + Google Maps APIs to standardize city/country names.
- Handled multilingual entries via Google Translate API (with manual validation).

MACS 30112

Winter 25

Final Project

Data cleaning n Visualization

Baihui Wang

Initial Data

	Repository	Stars	Contributo	Location
2	AutoGPT	171829	Auto-GPT-Bot	
3	AutoGPT	171829	Pwuts	Delft, NL
4	AutoGPT	171829	waynehamadi	
5	AutoGPT	171829	Torantulinc	UK
6	AutoGPT	171829	Swiftyos	
7	AutoGPT	171829	hunteraraujo	
8	AutoGPT	171829	richbeales	Kent, UK
9	AutoGPT	171829	majdyz	Amsterdam
10	AutoGPT	171829	aarushik93	Berlin
11	AutoGPT	171829	ntindle	Texas
12	AutoGPT	171829	BillSchumacher	
13	AutoGPT	171829	SilenNaihin	
14	AutoGPT	171829	dependabot[bot]	
15	AutoGPT	171829	Bentlybro	Some Where In Space I Think?
16	AutoGPT	171829	kcze	Scotland
17	AutoGPT	171829	lc0rp	
18	AutoGPT	171829	collijk	United States
19	AutoGPT	171829	AndresCd	Colombia
20	AutoGPT	171829	p-i-	Oxford, England
21	AutoGPT	171829	sadmuphir	Somewhere
22	AutoGPT	171829	maiko	Serris
23	AutoGPT	171829	drikusroor	Utrecht, Netherlands
24	AutoGPT	171829	Abhi19920	India
25	AutoGPT	171829	erik-megai	Denver
26	AutoGPT	171829	Wladastic	Germany
27	AutoGPT	171829	coditamar	
28	AutoGPT	171829	k-boikov	Sofia, Bulgaria
29	AutoGPT	171829	sweetlilmre	
30	AutoGPT	171829	csolar	
31	AutoGPT	171829	rihp	Schengen
32	AutoGPT	171829	Tymec	
33	AutoGPT	171829	kinance	
34	AutoGPT	171829	dschonhol	Boston, MA

20008	quadratic	3263	wavemind	Denver, CO
20009	quadratic	3263	golok727	✹
20010	quadratic	3263	juanbercoff	
20011	quadratic	3263	PawelJasti	Poznań | Nowy Tomyśl
20012	quadratic	3263	tmiller	
20013	quadratic	3263	boaarmpit	Tokyo
20014	quadratic	3263	ajboots	
20015	quadratic	3263	bibhubhatta	
20016	quadratic	3263	knewter	
20017	quadratic	3263	lTyl	Edmonton, Alberta
20018	quadratic	3263	KieranHolroyd	
20019	quadratic	3263	Nerptastic	
20020	quadratic	3263	N-Demir	
20021	quadratic	3263	manchegoP	

Data Cleaning

```
import pandas as pd
import numpy as np
import time
import re
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
import googlemaps

file_path = "/Users/wangbaihui/contributors_locations.csv"
print("Loading dataset...")
df = pd.read_csv(file_path)
print(f"Dataset loaded: {df.shape[0]} rows, {df.shape[1]} columns")

print("Dropping rows with missing locations...")
df_cleaned = df.dropna(subset=["Location"]).copy()
print(f"Remaining rows after drop: {df_cleaned.shape[0]}")
```

Extract cities from locations:

Example:

Location	Extracted City
"San Francisco, USA"	"San Francisco"
"Berlin, Germany"	"Berlin"
"Japan"	NaN

Extract countries from locations:

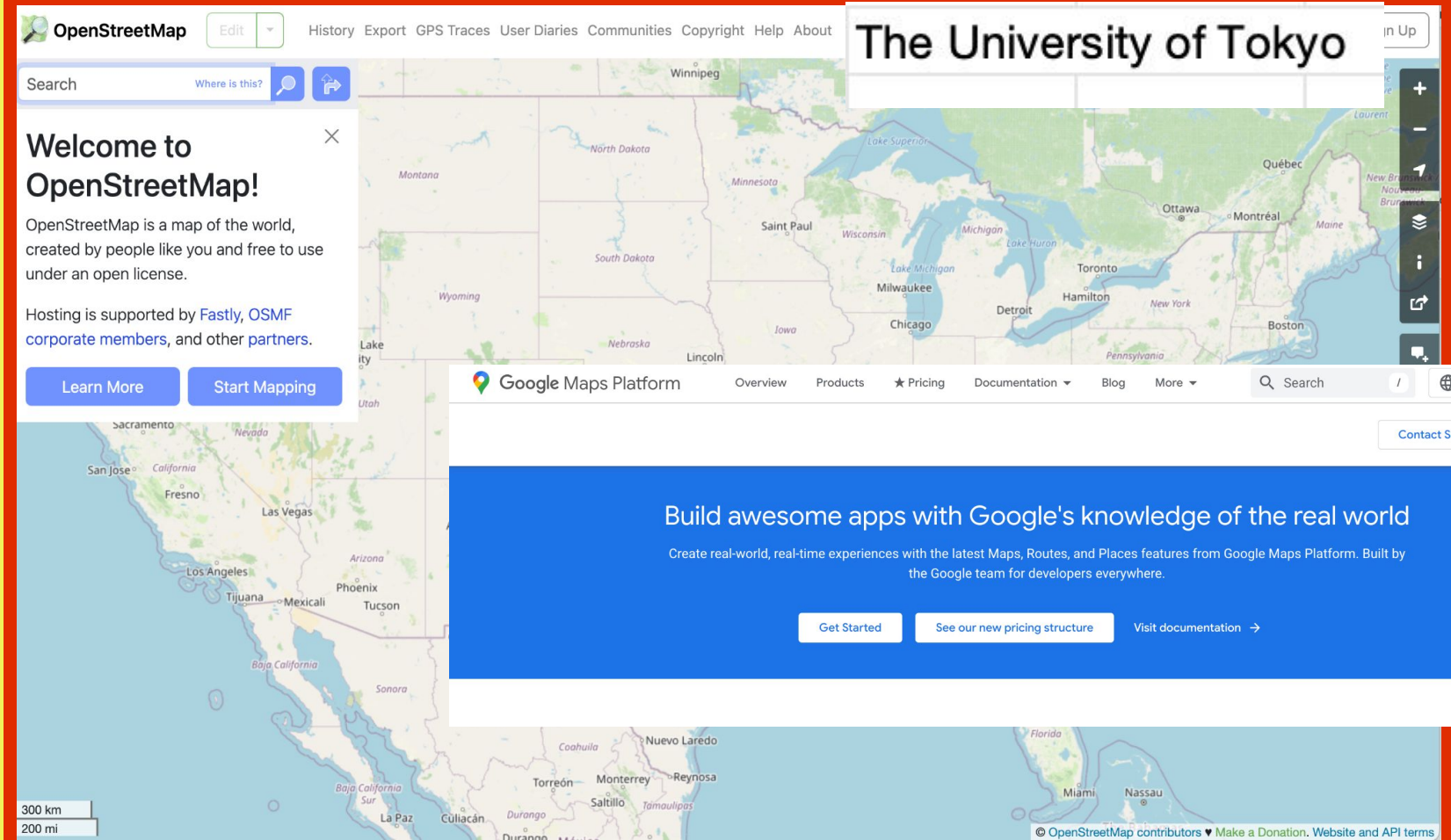
Example:

Location	Extracted Country
"San Francisco, USA"	"USA"
"Berlin, Germany"	"Germany"
"Japan"	"Japan"

AutoGPT	171829	Auto-GPT-Bot		
AutoGPT	171829	Pwuts	Delft, NL	
AutoGPT	171829	waynehamadi		
AutoGPT	171829	Torantulinc	UK	
AutoGPT	171829	Swiftyos		
AutoGPT	171829	hunteraraujo		
AutoGPT	171829	richbeales	Kent, UK	
AutoGPT	171829	majdyz	Amsterdam	
AutoGPT	171829	aarushik93	Berlin	
AutoGPT	171829	ntindle	Texas	

AutoGPT	171829	Pwuts	Delft, NL	Delft	Netherlands
AutoGPT	171829	Torantulinc	UK	Ouaka	Ködörösêse tî Bêafrîka / République centrafricaine
AutoGPT	171829	richbeales	Kent, UK	Kent	United Kingdom
AutoGPT	171829	majdyz	Amsterdar	Amsterdar	Nederland
AutoGPT	171829	aarushik93	Berlin	Berlin	Deutschland
AutoGPT	171829	ntindle	Texas	Texas	United States

Geocoding—handle partial location input



Geocoding—handle partial location input

```
cache_file = "geocode_cache.json"
if os.path.exists(cache_file):
    with open(cache_file, "r") as f:
        geocode_cache = json.load(f)
else:
    geocode_cache = {}

def get_lat_lon_batch(locations, batch_num):
    """Geocode a batch of locations using Google Maps API with caching and error handling."""
    batch_results = []
    locations_to_geocode = [loc for loc in locations if loc and loc not in geocode_cache]

    print(f"Processing Batch {batch_num}: {len(locations_to_geocode)} new locations...")

    if not locations_to_geocode:
        print(f"Batch {batch_num}: All locations already cached. Skipping.")
        return [geocode_cache.get(loc, (None, None)) for loc in locations]

    try:
        for loc in locations_to_geocode:
            response = gmaps.geocode(loc)
            if response:
                lat = response[0]['geometry']['location']['lat']
                lon = response[0]['geometry']['location']['lng']
                geocode_cache[loc] = (lat, lon)
                print(f"Geocoded: {loc} → ({lat}, {lon})")
            else:
                geocode_cache[loc] = (None, None)
                print(f"No geocode result for: {loc}")

            time.sleep(0.1)


        with open(cache_file, "w") as f:
            json.dump(geocode_cache, f)

    except Exception as e:
        print(f"Error in Batch {batch_num}: {e}")
        return [(None, None)] * len(locations)

    print(f"Batch {batch_num} completed!")
    return [geocode_cache.get(loc, (None, None)) for loc in locations]

batch_size = 10
total_batches = len(df) // batch_size + 1

for batch_num, i in enumerate(range(0, len(df), batch_size), start=1):
    df.loc[i:i+batch_size-1, ["Latitude", "Longitude"]] = get_lat_lon_batch(df["Location"].iloc[i:i + batch_size], batch_num)

df.dropna(subset=["Latitude", "Longitude"], inplace=True)

df.to_csv("geocoded_data.csv", index=False)
print("Geocoded data saved to geocoded_data.csv")
```

First-time geocoded data

Repository	Stars	Contributo	Location	City	Country
AutoGPT	171829	Pwuts	Delft, NL	Delft	Netherlands
AutoGPT	171829	Torantulinc	UK	Ouaka	Ködörösêse tî Bêafrîka / République centrafricaine
AutoGPT	171829	richbeales	Kent, UK	Kent	United Kingdom
AutoGPT	171829	majdyz	Amsterdar	Amsterdar	Nederland
AutoGPT	171829	aarushik93	Berlin	Berlin	Deutschland
AutoGPT	171829	ntindle	Texas	Texas	United States
AutoGPT	171829	Bentlybro	Some Whe	Unknown	Unknown
AutoGPT	171829	kcze	Scotland	Glasgow	Scotland
AutoGPT	171829	collijk	United Sta	United Sta	Unknown
AutoGPT	171829	AndresCd	Colombia	Colombia	Unknown
AutoGPT	171829	p-i-	Oxford, En	Oxford	England
AutoGPT	171829	sadmuphir	Somewher	Somewher	United Kingdom
AutoGPT	171829	maiko	Serris	Serris	France
AutoGPT	171829	drikusroor	Utrecht, N	Utrecht	Netherlands
AutoGPT	171829	Abhi19920	India	India	Unknown
AutoGPT	171829	erik-megar	Denver	Denver	United States
AutoGPT	171829	Wladastic	Germany	Deutschlar	Unknown
AutoGPT	171829	k-boikov	Sofia, Bulg	Sofia	Bulgaria
AutoGPT	171829	rihp	Schengen	Schengen	Lëtzebuerg
AutoGPT	171829	dschonhol	Boston, M	Boston	MA
AutoGPT	171829	Taytay	Tulsa, OK	Tulsa	OK
AutoGPT	171829	andrewhoc	Texas	Texas	United States
AutoGPT	171829	AlrikOlson	Portland, (	Portland	OR
AutoGPT	171829	OmriGM	Israel	לארשי	Unknown
AutoGPT	171829	Androbin	Passau, G	Passau	Germany
AutoGPT	171829	russellocea	South Car	South Car	United States
AutoGPT	171829	bituq	The Nethe	Nederland	Unknown
AutoGPT	171829	nponeccop	Barcelona,	Barcelona	Spain
AutoGPT	171829	eltociear	Tokyo, Jap	Tokyo	Japan
AutoGPT	171829	didier-dura	Switzerlan	Consulat G	France
AutoGPT	171829	Sma-Das	New York	City of Ne	United States
AutoGPT	171829	gersh	San Franci	San Franci	Canada

Lessons Learned & Optimization

Using batch geocoding significantly increases the efficiency and reduces processing time

```python
def get_lat_lon_batch(locations, batch_num):
    """Geocode a batch of locations using Google Maps API with caching and error handling."""
    batch_results = []
    locations_to_geocode = [loc for loc in locations if loc and loc not in geocode_cache]

    print(f"Processing Batch {batch_num}: {len(locations_to_geocode)} new locations...")

    if not locations_to_geocode:
        print(f"Batch {batch_num}: All locations already cached. Skipping.")
        return [geocode_cache.get(loc, (None, None)) for loc in locations]

    try:
        for loc in locations_to_geocode:
            response = gmaps.geocode(loc)
            if response:
                lat = response[0]['geometry']['location']['lat']
                lon = response[0]['geometry']['location']['lng']
                geocode_cache[loc] = (lat, lon)
                print(f"Geocoded: {loc} → ({lat}, {lon})")
            else:
                geocode_cache[loc] = (None, None)
                print(f"No geocode result for: {loc}")

            time.sleep(0.1)


        with open(cache_file, "w") as f:
            json.dump(geocode_cache, f)

    except Exception as e:
        print(f"Error in Batch {batch_num}: {e}")
        return [(None, None)] * len(locations)

    print(f"Batch {batch_num} completed!")
    return [geocode_cache.get(loc, (None, None)) for loc in locations]

batch_size = 10
total_batches = len(df) // batch_size + 1

for batch_num, i in enumerate(range(0, len(df), batch_size), start=1):
    df.loc[i:i+batch_size-1, ["Latitude", "Longitude"]] = get_lat_lon_batch(df["Location"].iloc[i:i + batch_size], batch_num)
```

```python
    except Exception as e:
        print(f"Error in Batch {batch_num}: {e}")
        return [(None, None)] * len(locations)

    print(f"Batch {batch_num} completed!")
    return [geocode_cache.get(loc, (None, None)) for loc in locations]

batch_size = 10
total_batches = len(df) // batch_size + 1

for batch_num, i in enumerate(range(0, len(df), batch_size), start=1):
    df.loc[i:i+batch_size-1, ["Latitude", "Longitude"]] = get_lat_lon_batch(df["Location"].iloc[i:i + batch_size], batch_num)

df.dropna(subset=["Latitude", "Longitude"], inplace=True)

df.to_csv("geocoded_data.csv", index=False)
print("Geocoded data saved to geocoded_data.csv")
```

Second-time geocoded data

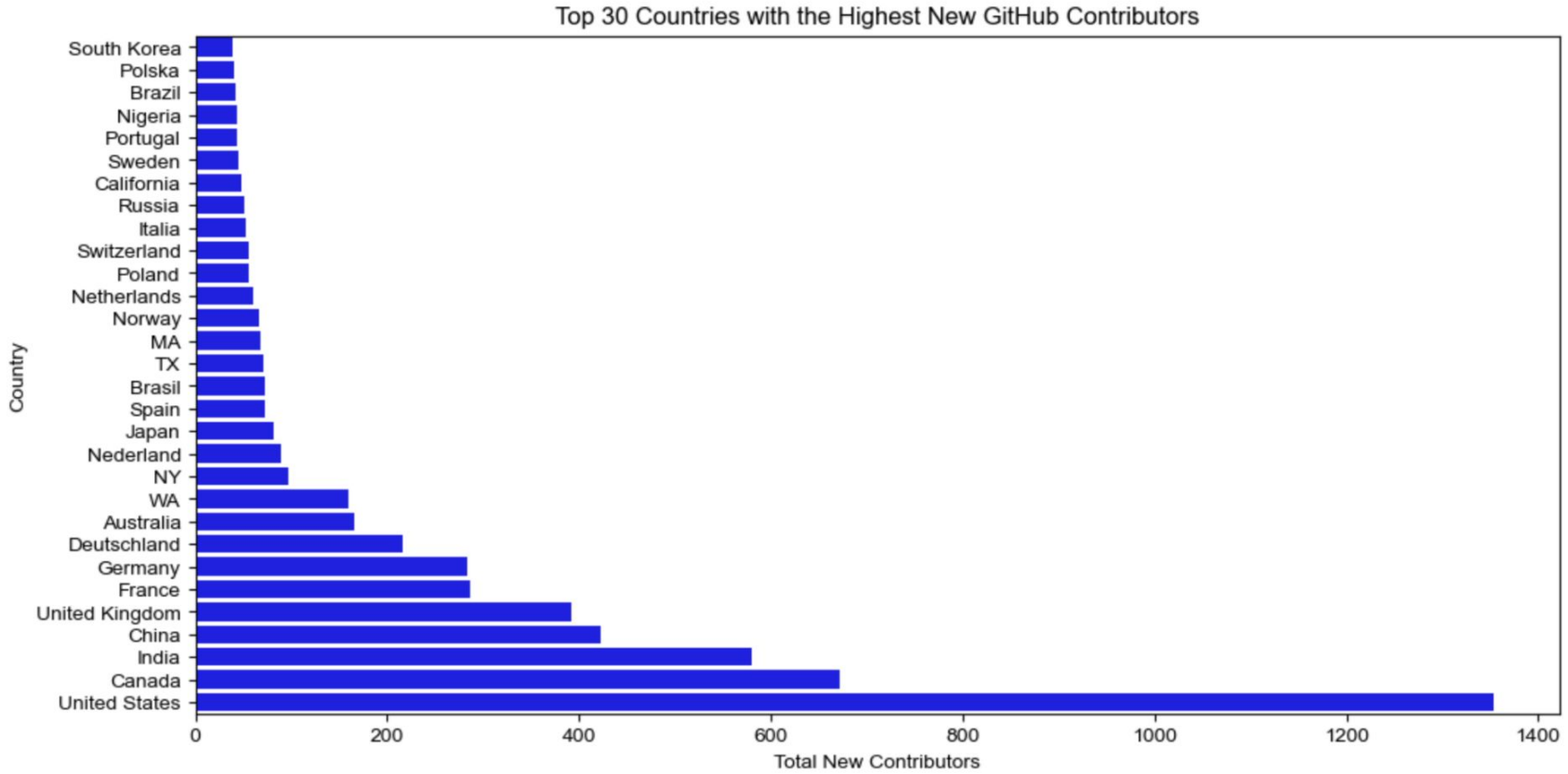
A	B	C	D	E	F	G	H	I	J	K	L
Repository	Stars	Contributo	Location	City	Country	Latitude	Longitude				
AutoGPT	171829	Pwuts	Delft, NL	Delft	Netherlanc	52.011576	4.357068				
AutoGPT	171829	Torantulinc	UK	Ouaka	Ködörösês	55.37805	-3.43597				
AutoGPT	171829	richbeales	Kent, UK	Kent	United Kin	51.09283	0.8392470999999999				
AutoGPT	171829	majdyz	Amsterdar	Amsterdar	Nederland	52.36757	4.904139				
AutoGPT	171829	aarushik93	Berlin	Berlin	Deutschlar	52.520006	13.40495				
AutoGPT	171829	ntindle	Texas	Texas	United Sta	30.272728	-97.7411				
AutoGPT	171829	kcze	Scotland	Glasgow	Scotland	56.490671	-4.20265				
AutoGPT	171829	collijk	United Sta	United Sta	Unknown	38.7946	-106.535				
AutoGPT	171829	AndresCd	Colombia	Colombia	Unknown	4.570868	-74.2973				
AutoGPT	171829	p-i-	Oxford, En	Oxford	England	51.75202	-1.25773				
AutoGPT	171829	maiko	Serris	Serris	France	48.84432	2.788231				
AutoGPT	171829	drikusroor	Utrecht, N	Utrecht	Netherlanc	52.091925	5.122957				
AutoGPT	171829	Abhi19920	India	India	Unknown	20.59368	78.96288				
AutoGPT	171829	erik-megal	Denver	Denver	United Sta	39.73924	-104.99				
AutoGPT	171829	Wladastic	Germany	Deutschlar	Unknown	51.16569	10.45153				
AutoGPT	171829	k-boikov	Sofia, Bulg	Sofia	Bulgaria	42.69771	23.32187				
AutoGPT	171829	dschonhol	Boston, M	Boston	MA	42.35551	-71.0565				
AutoGPT	171829	Taytay	Tulsa, OK	Tulsa	OK	36.15514	-95.9895				
AutoGPT	171829	andrewhoc	Texas	Texas	United Sta	30.272728	-97.7411				
AutoGPT	171829	AlrikOlson	Portland, C	Portland	OR	45.51523	-122.678				
AutoGPT	171829	OmriGM	Israel	לארשי	Unknown	31.04605	34.85161				
AutoGPT	171829	Androbin	Passau, G	Passau	Germany	48.56674	13.43195				
AutoGPT	171829	russellocea	South Car	South Car	United Sta	34.84849	-82.4				
AutoGPT	171829	bituq	The Nethe	Nederland	Unknown	52.13263	5.291265999999999				
AutoGPT	171829	nponeccor	Barcelona,	Barcelona	Spain	41.3874	2.168568				
AutoGPT	171829	eltociear	Tokyo, Jap	Tokyo	Japan	35.67642	139.65				
AutoGPT	171829	didier-dura	Switzerlan	Consulat C	France	46.81819	8.227511999999999				
AutoGPT	171829	Sma-Das	New York	City of Ne	United Sta	40.71278	-74.006				
AutoGPT	171829	gersh	San Franci	San Franci	Canada	37.77493	-122.419				
AutoGPT	171829	sagarisher	Åland	Åland	Suomi / Fi	60.17852	19.91561				
AutoGPT	171829	dillweed	USA	United Sta	Unknown	38.7946	-106.535				

Lessons Learned & Optimization

Interactive Mapping & Insights

https://github.com/macs30112-winter25/final-project-chick-fil-a/blob/main/Data%20Visualization/top_1000_contributors_map.html

Further Analysis



MACS 30112

Winter 25

Final Project

Data Wrangling

Typical errors

```
1   Country,total_contributors,total_stars,avg_stars_per_contributor
6   02139,1,5832,5832.0
7   059817,1,36520,36520.0
8   102676,1,71195,71195.0
9   110078,1,9536,9536.0
10  230022,1,12510,12510.0
11  29880,1,8023,8023.0
12  37.710840,1,35079,35079.0
13  53154,1,40041,40041.0
14  60637,1,11552,11552.0
15  65000,1,3448,3448.0
```

1. containing **numbers or symbols instead of actual country names**

.

Typical errors

```
1    Country,total_contributors,total_stars,avg_stars_per_contributor
504  Unknown,2133,65985851,30935.701359587434
505  Uruguay,3,177554,59184.666666666664
506  Uruguay.,1,9466,9466.0
```

2. Country names failing to merge correctly due to **extra symbols...**

```
  1  Country,total_contributors,total_stars,avg_stars_per_contributor
479  The  Netherlands,1,40203,40203.0
480  The Netherlands,22,865609,39345.86363636364
```

```
  1  Country,total_contributors,total_stars,avg_stars_per_contributor
534  West Bengal,2,27171,13585.5
535  WestBengal,1,9536,9536.0
```

Typical errors

2. Country names failing to merge correctly due to **extra symbols**

Or multiple spaces...

```
1    Country,total_contributors,total_stars,avg_stars_per_contributor
589  Česko,20,662714,33135.7
590  Ελλάς,10,305596,30559.6
591  Κύπρος - Kıbrıs,7,242829,34689.857142857145
592  България,5,183360,36672.0
593  Кыргызстан,1,19570,19570.0
594  Россия,43,1606794,37367.3023255814
595  Санкт-Петербург,1,4536,4536.0
596  Србија,6,95379,15896.5
597  Україна,10,185262,18526.2
598  Қазақстан,3,25688,8562.666666666666
599  Հայաստան,9,82476,9164.0
600  ישראל,10,366669,36666.9
601  الأردن,2,23624,11812.0
602  الإمارات العربية المتحدة,13,175154,13473.384615384615
603  البحرين,1,77961,77961.0
604  اليمن,1,41513,41513.0
605  ایران,12,442055,36837.916666666664
606  تونس,7,107792,15398.857142857143
607  لبنان,1,4206,4206.0
608  مصر,3,20762,6920.666666666667
609  موريتانيا,3,124427,41475.666666666664
610  پاکستان,11,555376,50488.72727272727
611  नेपाल,1,40041,40041.0
612  বাংলাদেশ,2,16086,8043.0
613  ประเทศไทย,5,133648,26729.6
614  საქართველო,1,8101,8101.0
615  ኢትዮጵያ إثيوبيا,1,29022,29022.0
616  中国,660,21432648,32473.70909090909
617  帝都,1,56047,56047.0
618  日本,66,2259715,34238.106060606064
619  臺灣,10,302257,30225.7
620  대한민국,31,1207980,38967.096774193546
```

Typical errors

3. Country names appearing in **different languages**.

Typical errors

```
  1  Country,total_contributors,total_stars,avg_stars_per_contributor
555  anyone can find me,1,11633,11633.0
556  az,1,4206,4206.0
557  but open to relocate,1,36153,36153.0
```

4. Or, some **naughty (misleading) words from user**

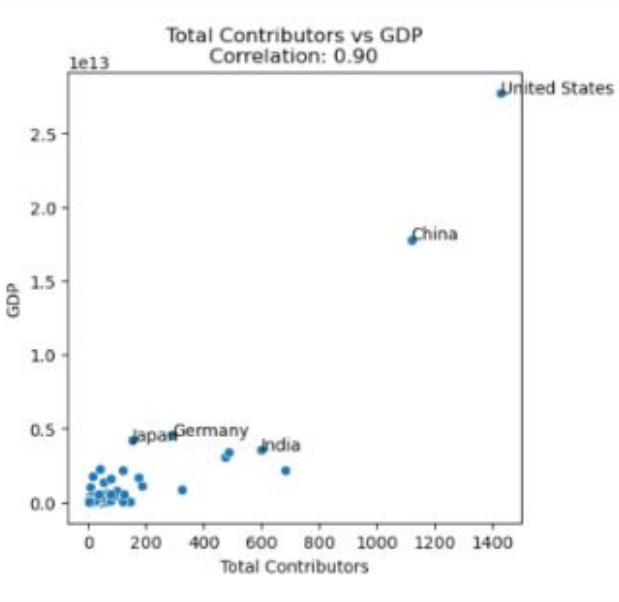
Regular Expressions;
Google Translator;
Pycountry;
high-threshold fuzzy matching;
Manual verification

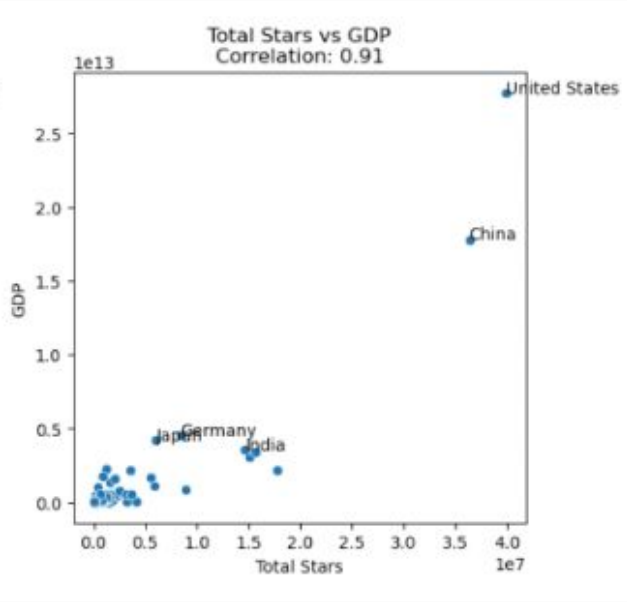
Oue new dataframe included:

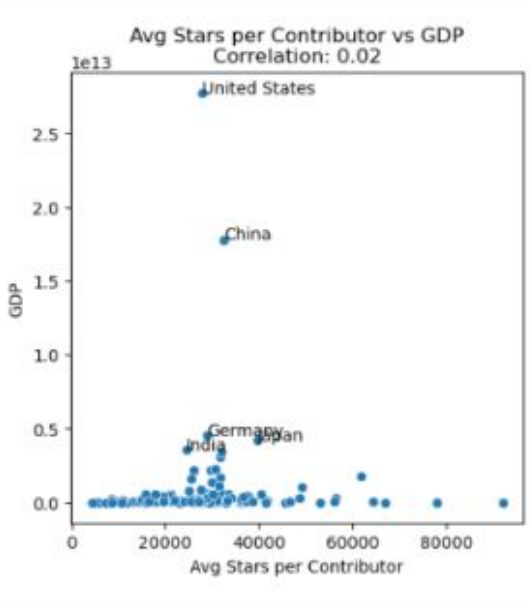
- **GDP per capita (2023);**
- **R&D Investment Index;**

  ——national investment into research and development

- **Annual AI Publications**







MACS 30112

Winter 25

Final Project

Data Analysis

Dependent Variable:

AI Open-Source Participation Index (AIOSPI)

1. **Geographic AIOSPI** (Total influence in a country):

$$AIOSPI = 0.5 \times (\text{number of contributors}) + 0.5 \times (\text{total star count})$$

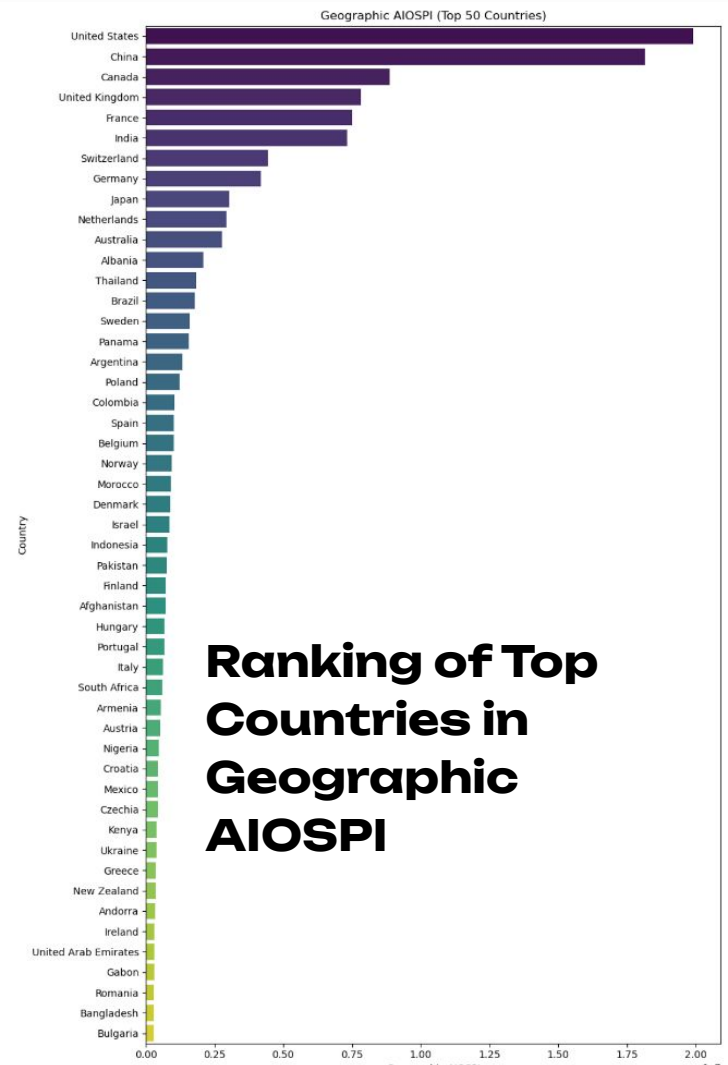
2. **Per Capita AIOSPI** (Influence per person):

$$AIOSPI = 0.5 \times \left(\frac{\text{number of contributors}}{\text{population}}\right) + 0.5 \times \left(\frac{\text{total star count}}{\text{population}}\right)$$

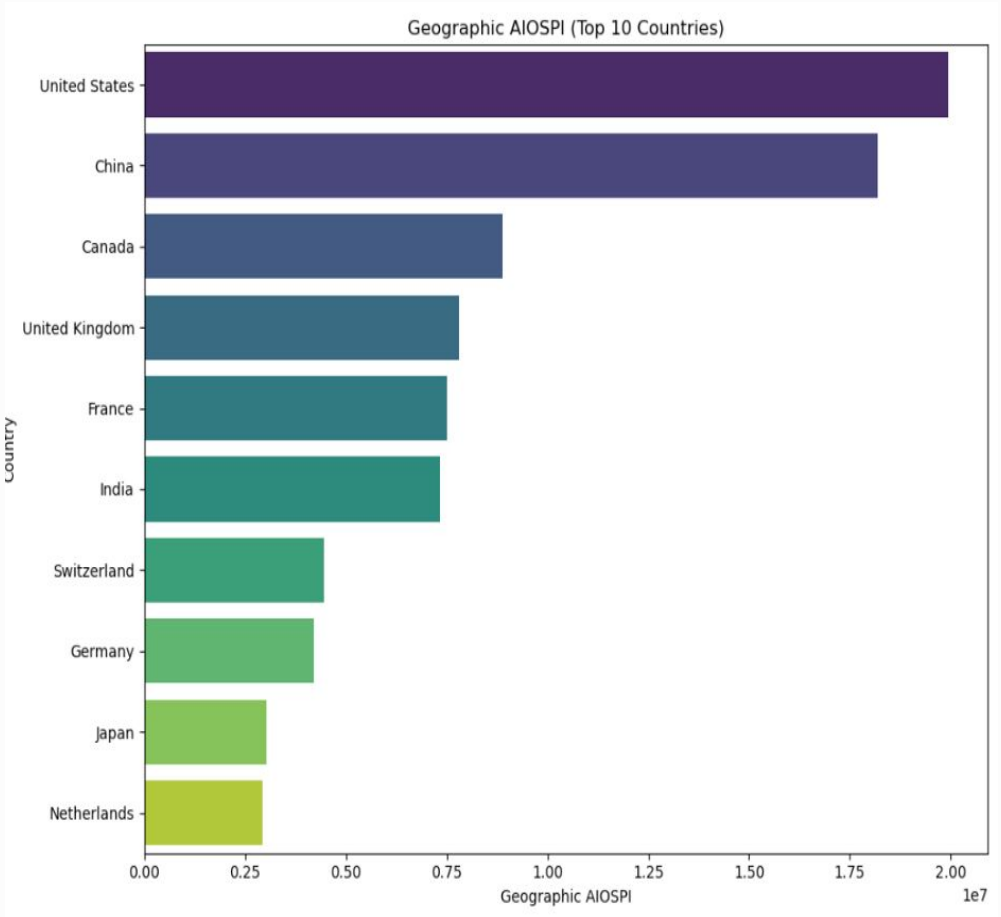
	total_contributors	total_stars	avg_stars_per_contributor
count	122.000000	1.220000e+02	122.000000
mean	68.803279	2.007207e+06	29167.534867
std	191.337685	5.605192e+06	15873.368717
min	1.000000	4.206000e+03	4206.000000
25%	2.250000	5.857075e+04	19092.935268
50%	10.500000	3.296325e+05	27834.798036
75%	44.000000	1.309785e+06	35494.525694
max	1426.000000	3.986350e+07	92111.500000

	Geographic_AIOSPI	Per_Capita_AIOSPI
count	1.220000e+02	122.000000
mean	1.003638e+06	0.129959
std	2.802691e+06	0.460559
min	2.103500e+03	0.000093
25%	2.928600e+04	0.003348
50%	1.648195e+05	0.020317
75%	6.549148e+05	0.067789
max	1.993246e+07	4.226038

	Population	star_per_capita	contributor_per_capita
count	1.220000e+02	122.000000	1.220000e+02
mean	5.497473e+07	0.259910	8.164675e-06
std	1.863675e+08	0.921091	2.742397e-05
min	1.187500e+04	0.000186	2.804868e-08
25%	3.408658e+06	0.006695	2.333296e-07
50%	1.075445e+07	0.040633	1.486054e-06
75%	3.642859e+07	0.135573	5.212759e-06
max	1.438070e+09	8.451803	2.720887e-04

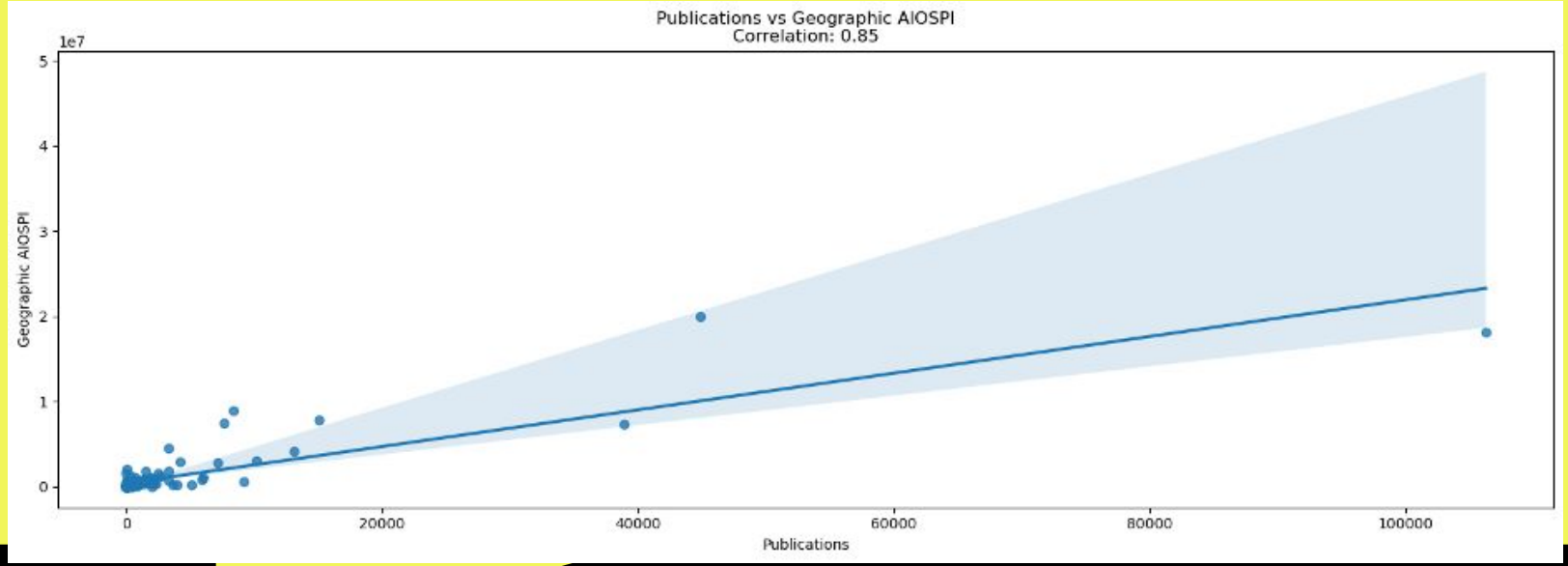


Ranking of Top Countries in Geographic AIOSPI



Correlations:

Geographic AIOSPI and AI Publications (Education/Research) had a **strong correlation ——0.85**



Regression Analysis:

OLS Regression Results

Dep. Variable:	Geographic_AIOSPI	R-squared:	0.726
Model:	OLS	Adj. R-squared:	0.723
Method:	Least Squares	F-statistic:	306.6
Date:	Wed, 05 Mar 2025	Prob (F-statistic):	2.34e-34
Time:	08:10:10	Log-Likelihood:	-1844.2
No. Observations:	118	AIC:	3692.
Df Residuals:	116	BIC:	3698.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.907e+05	1.43e+05	2.740	0.007	1.08e+05	6.73e+05
Publications	215.5584	12.310	17.511	0.000	191.177	239.940

Omnibus:	125.136	Durbin-Watson:	1.668
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2354.561
Skew:	3.622	Prob(JB):	0.00
Kurtosis:	23.650	Cond. No.	1.20e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.2e+04. This might indicate that there are strong multicollinearity or other numerical problems.

1. Leading countries:

United States, China, Germany, India, and Japan

2. Leading cities:

San Francisco, Beijing, Seattle, London, New Delhi, Berlin, and Deutschland

Key Findings:

3. Important indicators:

GDP

high-level research output (AI publications)

4. Global Inequality

Thanks for your listening!

Anqi Wei, Charlotte Li, Baihui Wang