An analysis of in-country article references and predictive modeling of local ratios based on country neighbors

Victoria Dollar, Ben Galin, Tyler Karhoff, and Gregory Smith

Emporia State University

January 13, 2023

Introduction and Background

Citations are an important tool for recording the use of public research. They allow recognition to authors for their methodology and contribution to their field. Citation formatting grants scholars the means to trace sources shared throughout the literature in a more manageable way. With the invention of the world wide web, published articles have become more accessible at greater distances. As a result, global inquiries examine the geolocation of the author in reference to subject matter, country of origin, number of citations used, the number of times the article was cited, etc.

An egocentric network centralizes the relationship between an entity of interest and its subnetwork. [9] For a particular publication, a local ratio can be constructed comparing the countries of the author to the countries of the citations' authors used in that article. A higher ratio would implicate that more citations utilized, originated from its own country whereas a lower ratio would reflect the usage of articles stemming from other countries. This paper will focus on comparing the ratios of various countries of statistics journals and estimating a country's ratio based upon the ratios of its neighbors. In the end, the hope is to gain insight on the relationship of networks between countries in the world of academia.

One factor that limits the use of a journal in research is accessibility. Since the birth of the internet, the number of journal publications have increased. Brazil (18.6%), the United Kingdom (10.7%), and the United States (6.4%) were reported to have the highest input in open-access citations. [3] Open-access journals have contributed to this increase by allowing three-times more articles to be published, [5] and as a result, studies have been done to determine their impact. Antelman has reported that these journals have a much greater, yet complex impact on research. [2] It was found that the number of citations did not increase, but open-access articles were more favored. Similarly, Davis et al. notes that open-access allows for an increase in the number of readers but not necessarily the number of citations. [5] Although data will not be available to determine if open-access articles affect the local ratio, these articles could influence our results.

To date, no study has been published to determine the correlation between a country and its neighbors, in terms of academic references. However, one study was found that is similar to the study we performed. Gouel et al. examined the prevalence ratio, comparing the IP geolocations of readers to the IP geolocation of the author of online published articles. [6] The results showed that 98% of the webpage views derived from the country of origin. The mean prevalence ratio was 0.9 meaning that most of the articles did not network outside of their country. The minimum ratio was from Kossovo (0.18), and an unexpectedly low ratio of 0.6 was recorded for the United Kingdom. There were four additional studies that provided some insight into the number of citations used in articles in ocean acidification and fuzzy research studies by country.

Sahoo and Pandley tracked and mapped 100 published articles on ocean acidification. [7] The United States (57 citations), Germany (26 citations), and Australia (24 citations) were the top three countries that had their articles cited the most often. Alfaro-Garcia et al. published a study determining the total citation counts in fuzzy research articles for countries around the world, and the top three countries were China and Taiwan (141,298 citations), the United States (37,601 citations), and Iran (25,291 citations). [1]

Based upon this research, we can observe that these top countries are not located near one another. In effect, we can hypothesize that there will be no spatial correlation between a country's local ratio and the estimations produced from neighboring countries. Reasoning that the size, GDP, and commitment to education advancement will make a much larger impact with the number of publications and contribution to research over a country's geolocation.

Methods

Data collection

We started with an initial seed of the 100 most recent articles as of November 19, 2022, matching the query term `statistics` in Crossref, a DOI registration agency website. Identifying each article by its DOI, we then used Crossref's public API to retrieve metadata about the article. Specifically, we retrieved metadata about the authors and their affiliations, and about the reference DOIs used by each article. We then repeated the process for each discovered reference DOI, running enough iterations until more than 500,000 articles were fetched.

For about 70% of articles, at least one of the authors included an affiliation name, often the name and address of a university. We attempted to find the country associated with each affiliation by parsing this value and matching it against a list of country names. We were able to associate about 8% of articles with a country in such a manner.

Detailed documentation for this data collection process is located in the Appendix under Listing 2.

Local ratio computation

For each article associated with a country, we looked at its references and their associated countries. We then computed the *article local ratio* as the number of references from the same country divided by the total number of references. We disregarded references that we were unable to associate with a country.

As a concrete example, consider this very article. Of the references listed below, seven have DOIs. Of these seven, only two — [7] and [4] — have affiliation metadata listed in the Crossref database, which we can parse and find to be India and Norway, respectively. Thus, this article's local ratio is 0/2.

We then computed the *country local ratio* by taking the average of the article local ratios from this country. We are able to see these results in Figure 6 which was created using the R library tmap. [8]

Selecting neighbors

When selecting what constitutes a 'neighbor' for specific sovereign countries, we looked at three distinct methods. These methods were able to be performed using the R library spdep.[4] The first and perhaps most intuitive method is to say that neighbors are contiguous, or connected by a border. Doing so yields the result found in Figure 1.

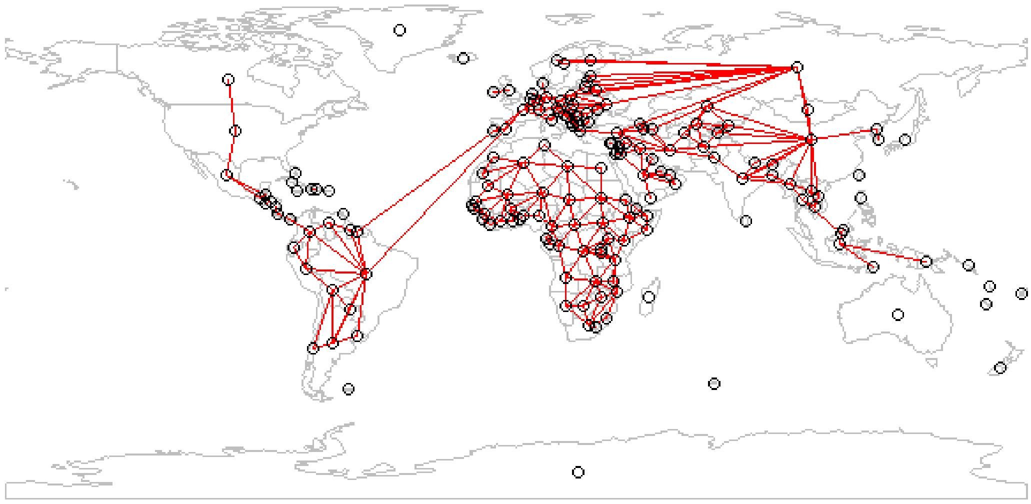

Figure 1: Boarder Neighbors (Red Line connects all Neighbors that Share a Border)

Notice that this is fairly accurate; however, it does have some drawbacks when it comes to analysis. For instance, countries with multiple sections cannot be well-matched. Notice that France is connected to Brazil and Suriname due to French Guiana being a part of France. This method's second problem is that island nations are all neighbor-less, making analysis with these countries impossible with this method. Thus, we will not be using this method for analysis, as it is too restrictive.

The second method we considered is using the centroids of all nations and a threshold maximal distance between them. There are many possible choices for a threshold. Two of them are shown in Figures 2 and 3.

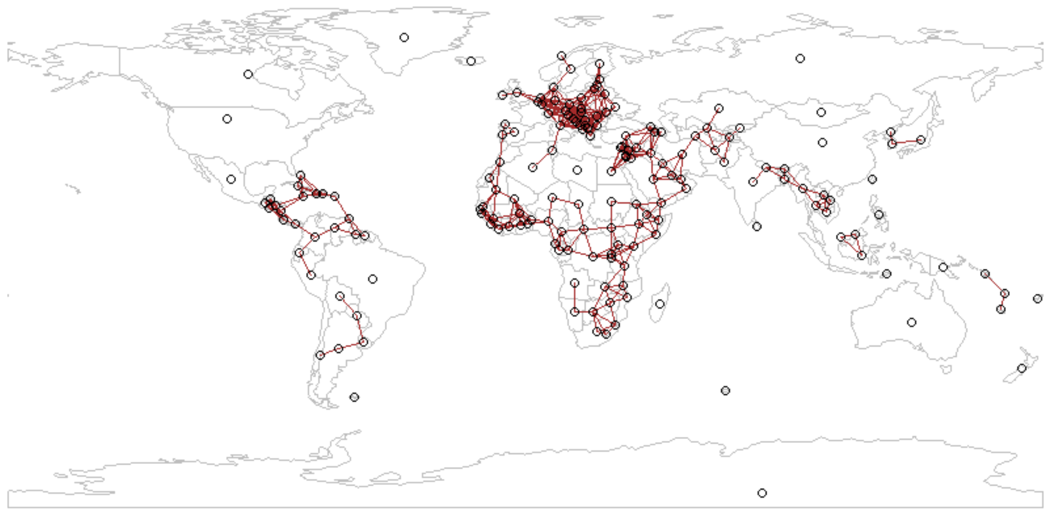

Figure 2: Centroid Neighbors (Brown Line connects all Neighbors that have close geographical centroids)

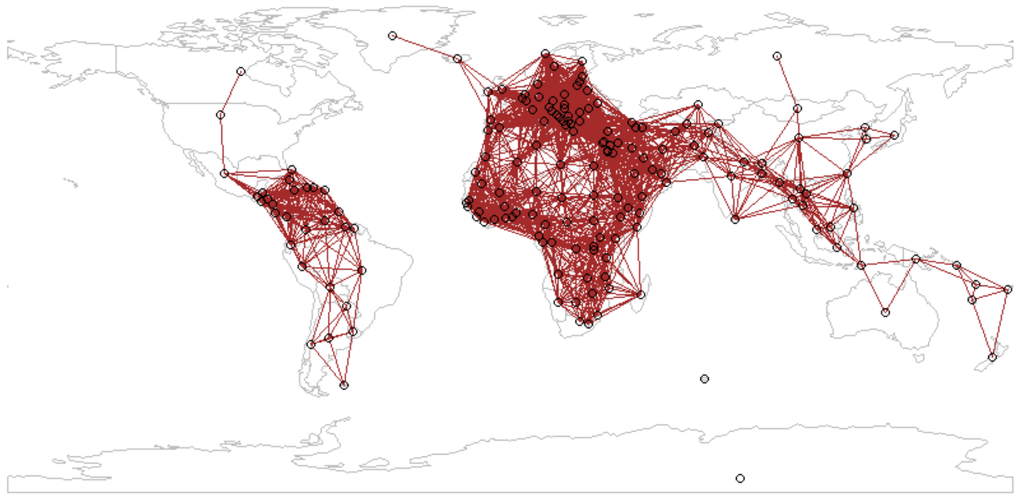

Figure 3: Centroid Neighbors (Brown Line connects all Neighbors that have further geographical centroids)

Both of these choices can cause problems with our data set. The first choice doesn't allow for much distance between neighbors and picks what countries are neighbors conservatively, while the second is much more liberal. We believe that the distinction in selecting neighbors with this method is also difficult to test properly, so we also have chosen not to use it. However, for a secondary study, it would be interesting to compare the results from this method and our last and chosen method.

The third method we considered is to use centroids as before, but instead of picking a maximal distance, we looked at a predefined number of nearest neighbors. As with the distance threshold approach, we experimented with different values for the number of nearest neighbors. Figures 4 and 5 show the results with three and five nearest neighbors.

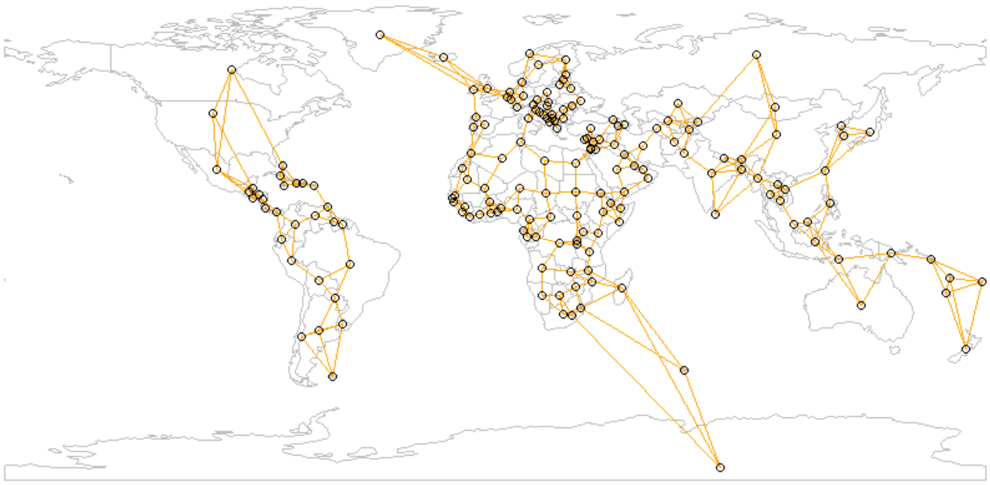

Figure 4: Centroid Neighbors (Orange Line connects three closest Neighbors)

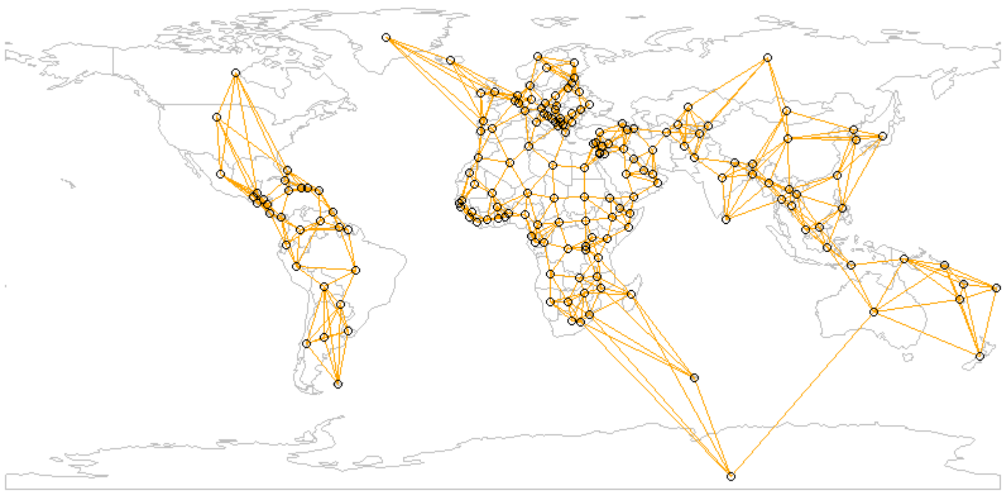

Figure 5: Centroid Neighbors (Orange Line connects five closest Neighbors)

The final method provides a pleasant result. We chose this method for our study for several reasons. First, unlike the first method, island nations can have neighbors for analysis. Second, varying the specific number of neighbors allows us to easily conduct several analyses: using a conservative choice for the number of neighbors and a more liberal choice for the number of neighbors when trying to create our predictive model. Our hope was to see which neighbor count provided the best prediction for local ratios. If a neighboring country was missing its local ratio, then we selected the next neighbor. We found this to be the best way to pass over unknown countries without them vastly altering our results.

Spatial autocorrelation

Once we had a method of selecting neighbors, our goal was to see if there is spatial autocorrelation present in the country's local ratios. We computed the Moran I statistics [4] with three, five, and ten neighbors. We then adjusted the weighting function of the neighbors to an Inverse Distance Weight (IDW) and repeated the test.

Further, we tried to predict countries' local ratios by examining the local ratios of the countries nearest to them. Again, we used both a simple moving average and an IDW-based moving average for every sampled country. For example, the United States' closest three neighbors with data points are Canada, Mexico, and Cuba. Thus, when calculating the estimated ratio for the United States, we took the mean and the weighted averages of these three results. We then computed the Pearson correlation coefficient, between our sample local ratios and the estimated ratios. We did this for several neighbor counts to see if any provided a correlation. The details are in our results and analysis, and the code implementation is documented in Listing 1.

Listing 1: R Implementation of IDW

```
library(sf)
library(magrittr)


idw = function(Countries_df) {
  World_centroids <- suppressWarnings(
    st_centroid(Countries_df) %>%st_coordinates()
  )

  corr_list <- list()
  for (m in 1 : 20) {
    # Find the m nearest neighbors based on the centroids of each country.
    World.knb <- knn2nb(knearneigh(World_centroids, k = m))

    # ‘country_sma‘ will hold the estimated local ratios, based on neighbors.
```

```
    country_sma <- list()

    # Iterate over countries.
    for (i in 1 : nrow(Countries_df)) {
      neighbor_list <- World.knb[[i]]

      # Compute the first-power IDW of the current country.
      # The formula is (\sum_j r_j/d_j)/(\sum_j 1/d_j), where
      #   r_j is the local ratio of neighbor j; and
      #   d_j is the centroid distance between neighbor j and our country i.
      numerator = 0
      denominator = 0
      for (j in 1 : m) {
        distance <- as.numeric(st_distance(
          suppressWarnings(st_centroid(Countries_df[i,])),
          suppressWarnings(st_centroid(Countries_df[neighbor_list[[j]],]))))
        local_ratio <- Countries_df[neighbor_list[[j]],]$local_ratio
        numerator <- numerator + local_ratio/distance
        denominator <- denominator + 1/distance
      }
      country_sma <- append(country_sma, numerator/denominator)
    }

    # Compute the Pearson correlation between the actual local_ratios and the
    # estimated ones.
    country_sma <- as.numeric(country_sma)
    knb_final_df <- data_frame(Countries_df$sovereignt, Countries_df$local_ratio, country_sma)
    colnames(knb_final_df) <- c('sovereignt','local_ratio', 'country_sma')
    corr_list <- append(corr_list, cor(knb_final_df$local_ratio, knb_final_df$country_sma))
  }

  return(corr_list)
}
```

Results & Analysis

The results located in Table 1 compare the sample local ratios with their estimations based on a simple moving average of the neighbors' local ratios. We are showing the results for three-, five-, and ten-closest neighbors, along with visualizations for each in figures 7, 8, and 9.

Country	Sample Local Rat.	3 Neigh Est.	5 Neigh Est.	10 Neigh Est.
Algeria	0.0000	0.1791	0.1688	0.1036
Argentina	0.1855	0.0738	0.0443	0.0681
Australia	0.1893	0.0095	0.0331	0.0991
Austria	0.1084	0.1396	0.1518	0.1140
Bangladesh	0.3333	0.1217	0.1031	0.1017
Belgium	0.1288	0.1073	0.1330	0.1606
Brazil	0.1007	0.0000	0.0612	0.0747
Canada	0.1658	0.3773	0.2430	0.1531
Chile	0.1206	0.0954	0.0572	0.0746
China	0.1505	0.1111	0.1412	0.1088
Colombia	0.0000	0.0278	0.1082	0.1622
Costa Rica	0.0000	0.1468	0.0919	0.1622
Croatia	0.0000	0.0842	0.0806	0.1219
Cuba	0.3571	0.0278	0.0205	0.1310
Cyprus	0.0000	0.1094	0.0870	0.0885
Czech Republic	0.3772	0.0500	0.0762	0.1025
Denmark	0.1535	0.1204	0.1735	0.1590
Egypt	0.0601	0.1094	0.0657	0.0675
Estonia	0.0156	0.1101	0.0960	0.1146
Finland	0.2020	0.0551	0.0959	0.1113
France	0.1563	0.0730	0.1699	0.1300
Georgia	0.2288	0.0356	0.0213	0.0666
Germany	0.1898	0.1073	0.1167	0.1282
Greece	0.1607	0.0342	0.0288	0.0887
Guinea	0.0000	0.1222	0.0869	0.1043
Hungary	0.1026	0.0139	0.1094	0.1156
Iceland	0.1667	0.1561	0.1537	0.1279
India	0.1291	0.2230	0.1810	0.1453
Indonesia	0.0000	0.0095	0.0803	0.1155
Iran	0.1511	0.0556	0.0950	0.1139
Iraq	0.0000	0.0547	0.0816	0.1061
Ireland	0.0484	0.1830	0.1576	0.1362
Israel	0.1642	0.0547	0.0449	0.0800
Italy	0.1503	0.0500	0.1355	0.1131
Japan	0.1759	0.0723	0.0735	0.0965
Jordan	0.0000	0.1094	0.0657	0.0970
Kenya	0.4097	0.0466	0.0613	0.0704
Latvia	0.0000	0.1153	0.0692	0.1084
Lebanon	0.0000	0.1094	0.0870	0.0964
Luxembourg	0.0000	0.1502	0.1588	0.1735
Malaysia	0.0286	0.0787	0.0472	0.1066
Luxembourg	0.0000	0.1502	0.1588	0.1735
Malaysia	0.0286	0.0787	0.0472	0.1066
Mexico	0.0189	0.3710	0.2393	0.1769
Morocco	0.3667	0.0569	0.0654	0.0805
Mozambique	0.0000	0.1596	0.1237	0.1114
Myanmar	0.0000	0.1898	0.1698	0.1350
Netherlands	0.1715	0.1062	0.1245	0.1564
Nigeria	0.0000	0.0000	0.0854	0.1318
Norway	0.1859	0.1684	0.1042	0.1516
Pakistan	0.3356	0.0934	0.1561	0.1089
Panama	0.0833	0.1190	0.0752	0.1539
Philippines	0.0000	0.0551	0.0803	0.1141
Poland	0.1282	0.1599	0.1176	0.0835
Portugal	0.0677	0.2087	0.1876	0.1389
Qatar	0.1667	0.0770	0.0462	0.1124
Romania	0.0000	0.0878	0.0783	0.0919
Russia	0.1436	0.1880	0.1799	0.1577
Saudi Arabia	0.0798	0.0556	0.0662	0.0706
Senegal	0.0000	0.1222	0.0869	0.1043
Slovakia	0.0000	0.2027	0.1299	0.1098
Slovenia	0.0417	0.1619	0.1477	0.1207
South Africa	0.0690	0.1366	0.0940	0.0880
South Korea	0.0801	0.1544	0.0926	0.1061
Spain	0.1030	0.1969	0.1740	0.1353
Sweden	0.1498	0.1805	0.1114	0.1424
Switzerland	0.1504	0.1062	0.1154	0.1321
Taiwan	0.1367	0.0267	0.0813	0.1005
Thailand	0.2361	0.1111	0.0997	0.0858
Turkey	0.1067	0.0547	0.0657	0.0718
Ukraine	0.0000	0.0356	0.0245	0.0743
United Kingdom	0.2637	0.1162	0.1010	0.1290
United Arab Emirates	0.0000	0.1325	0.1466	0.1290
United States	0.7558	0.1806	0.1250	0.1013
Uruguay	0.0000	0.1356	0.0814	0.0847
Venezuela	0.0000	0.0278	0.1082	0.0866
Vietnam	0.0000	0.0883	0.0803	0.1095

Table 1: Sample Ratios and Estimated Local Ratios for 3, 5, and 10 Neighbors

The estimates tend to be more accurate when the local ratio is around 0.10-0.20. Exceptionally high or exceptionally low local ratios are estimated poorly by their neighbors. This is an example of a "regression to the mean", as the neighbors' local ratios are less extreme.

Looking at the top ten countries based on local ratios, we didn't notice any obvious trends. These countries — United States, Kenya, Czech Republic, Morocco, Cuba, Pakistan, Bangladesh, United Kingdom, Thailand, and Georgia — include both large and small countries, countries from different continents, high GDP and low GDP countries, etc.

Using the Moran I test, we found no autocorrelation in the local ratios, irrespective of the number of neighbors and the average weighting function used. See Table 2.

Neighbors	Weight	p-value	Moran I statistic
3	simple average	0.673	−0.047723953
3	IDW	0.7221	−0.08322051
5	simple average	0.5844	−0.02575441
5	IDW	0.6503	−0.05582828
10	simple average	0.6703	−0.030992166
10	IDW	0.5325	−0.02120910

Table 2: Moran I test results

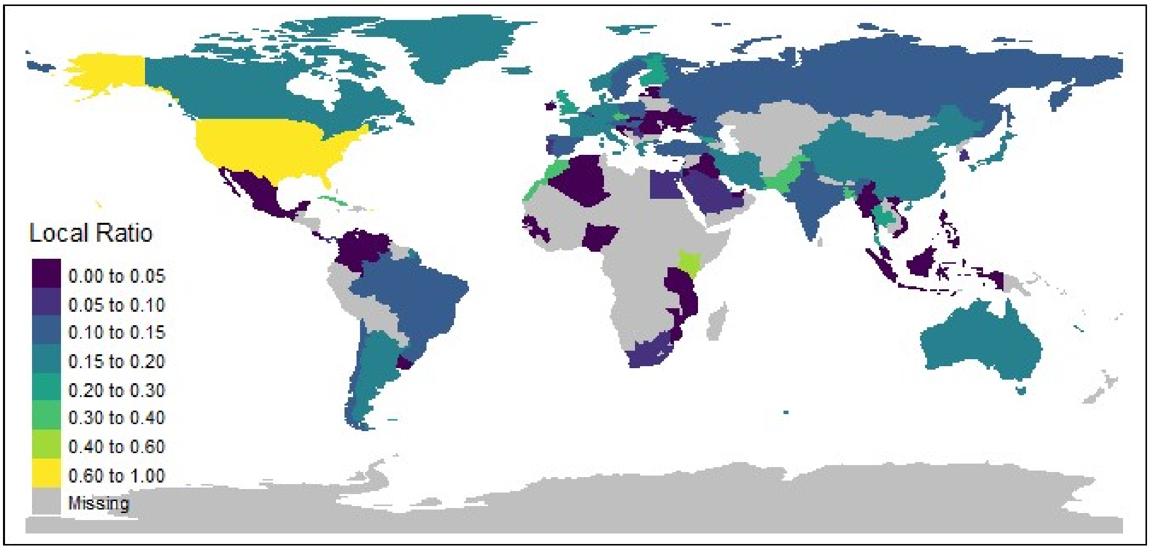

Figure 6: Local Ratio by Country (Breaks are not perfectly divided. This is present in all future maps)

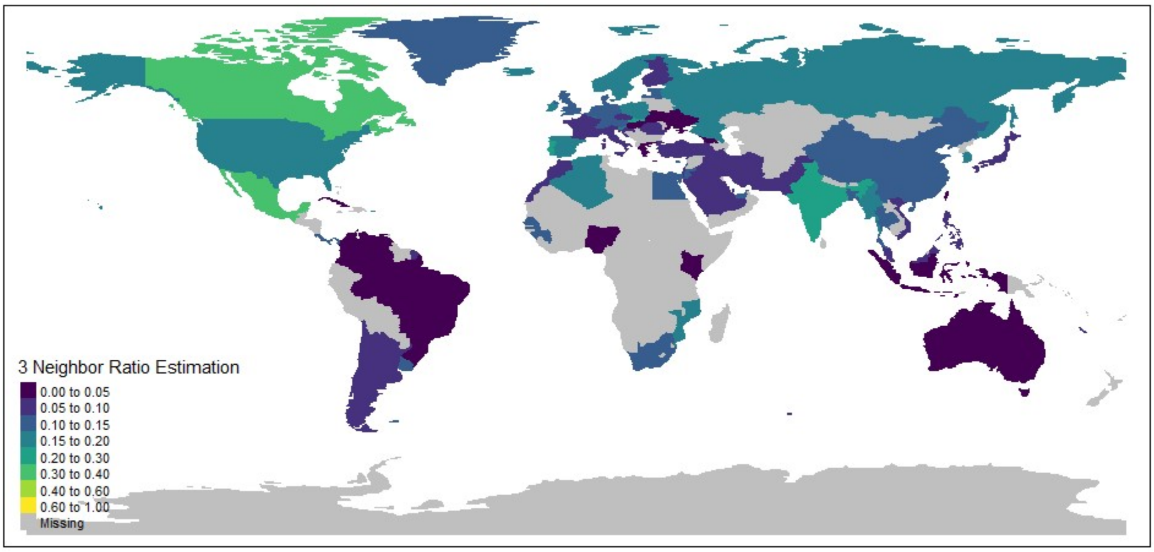

Figure 7: Estimating Local Ratio Based on 3 Closest Neighbors

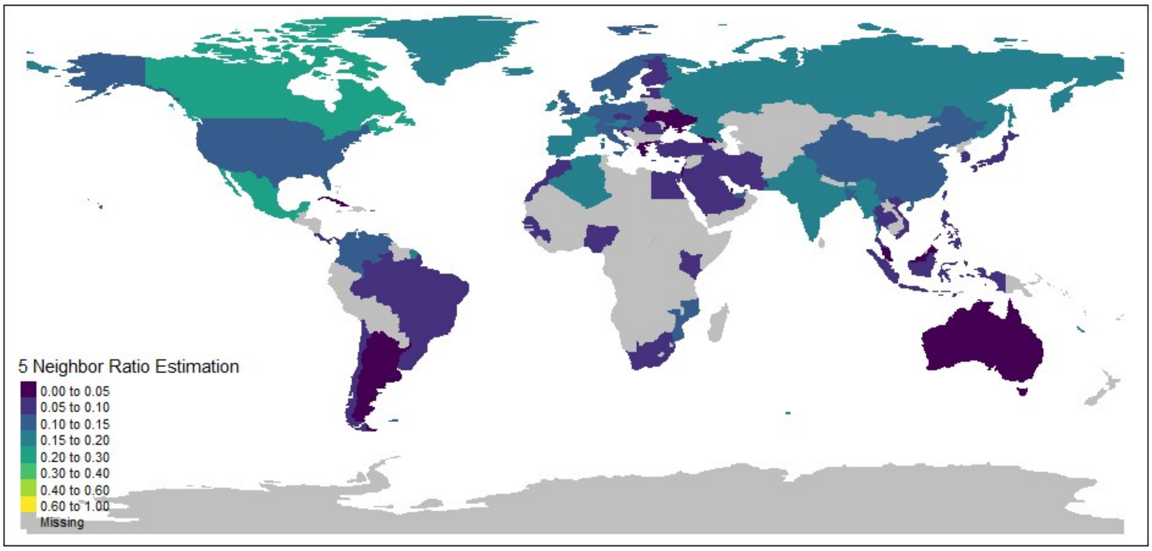

Figure 8: Estimating Local Ratio Based on 5 Closest Neighbors

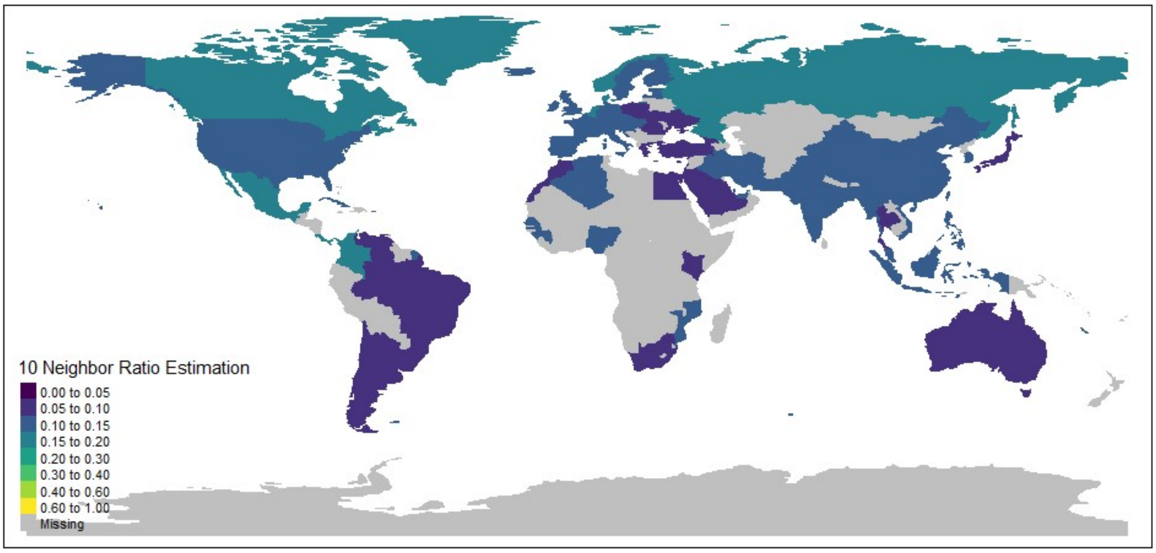

Figure 9: Estimating Local Ratio Based on 10 Closest Neighbors

The thematic maps [8] in Figures 7, 8, and 9 visualize the estimated local ratios for three-, five-, and ten-neighbors, respectively. The Pearson correlation coefficients are -0.08678 for three-, -0.07008 for five-, and -0.14110 for ten neighbors. When peering into more neighbors the Pearson coefficient stays negative with little correlation. Similarly, the Moran I test confirms no autocorrelation is present. This shows that the sample local ratio cannot be reliably estimated from the local ratios of neighboring countries.

Discussion & Conclusion

As we hypothesized, there is no correlation between a country's local ratio and the estimation of the neighboring countries. We can conclude that using k-neighbors, is not the best method of estimation per the results of the Moran I test. In respect to this, there are many contributing factors that could influence the lack of correlation, such as, the number of universities and other research facilities located in a country, the country's GDP and population, and geolocation in reference to other research-focused groups.

In addition, the process of matching an article to a country is imprecise. The affiliation data stored with each article is open-form, and authors often enter non-standard address information or location. Since we processed a large number of records, we had to automate this parsing and matching logic. In the case of United States-based articles, we were aided by our knowledge of common forms of address writing (for example, addresses ending with a state two-letter abbreviation and a 5- or 9-digit zip code). Presumably, we were thus able to detect a larger proportion of United States articles. This, in effect, contributed to a higher computed local ratio in the United States as compared to the true local ratio. Conversely, for countries with a large fraction of United States citations, we computed a lower local ratio than we would have had, had we not been able to parse United States addresses. A more sophisticated language parsing scheme, such as from natural language processing, would undoubtedly improve this result, which is an avenue of potential future study.

Another main contributor to the data quality is the sparse nature in which reference DOIs are present for an article given the different citation forms (APA, MLA, Chicago, AMS, etc.). Consequently, many articles do not have any importance in regard to analysis besides being a reference to source article. Ultimately, both of the above factors require retrieval of mass articles to gather a sufficient sample of articles that can be assigned to source countries. For a more representative sample, magnitudes greater than half a million articles should be amassed for ratio analysis.

References

[1] ALFARO-GARCIA, V., AND MERIGO, J. A citation analysis of fuzzy research by universities and countries. *Journal of Intelligent & Fuzzy Systems 38* (2020).

[2] ANTELMAN, K. Do open-access articles have a greater research impact? *College & Research Libraries 65*, 5 (2004).

[3] BHARDWAJ, R. India's contribution to open access movement. *Journal of Knowledge & Communication Management 5*, 2 (2015), 107–126.

[4] BIVAND, R. R packages for analyzing spatial data: A comparative case study with areal data. *Geographical Analysis 54*, 3 (2022), 488–518.

[5] DAVIS, P., LEWENSTEIN, B., AND ET AL. Open access publishing, article downloads, and citations: Randomised controlled trial. *BMJ* (2008).

[6] GOUEL, M., VERMEULEN, K., AND ET AL. Longitudinal study of an ip geolocation database. *arXiv* (2021).

[7] SAHOO, S., AND PANDEY, S. Characteristics and inter-citation network of 100 most influential studies on ocean acidification: A bibliometric analysis. *Science & Technology Libraries 41*, 1 (2022), 56–72.

[8] TENNEKES, M. tmap: Thematic maps in R. *Journal of Statistical Software 84*, 6 (2018), 1–39.

[9] WENG, T., LI, Z., AND ZHANG, J. Egocentric visual analysis of dynamic citation network. *Journal of Visualization 25* (2022), 1343–1360.

Appendices

Python Code

```python
import re
import os
import pandas as pd
import numpy as np
import time
from concurrent.futures import ThreadPoolExecutor, as_completed
from habanero import Crossref
import urllib.parse
import math
from typing import Set, Dict, List, Tuple


# ARTICLE RETRIEVAL #

def parse_dois(article: dict) -> Set[str]:
    return set([urllib.parse.unquote(ref.get('DOI').strip()) for ref in article.get('reference', []) if ref.get('DOI')])


def fetch_batch_to_df_threaded(dois: List[str], max_articles=100, max_iteration=3,
        runner_sleep=0.1, max_workers=20) -> pd.DataFrame:
    articles_df = pd.DataFrame()
    start_time = time.time()
    iteration = 0
    while (articles_df.shape[0] < max_articles) and (len(dois) > 0):
        iteration += 1
        # boolean that determines if we get next `dois`
        # on final iteration is `False` so len(dois) == 0 and loop breaks
        get_next = iteration < max_iteration
        dois, articles_df = fetch_threaded(dois, articles_df, start_time, get_next, runner_sleep, max_workers, max_articles)
        print('== end of iteration #%s (%d articles) (%d batches next iteration) (%.2fs) =='
            % (iteration, articles_df.shape[0], math.ceil(len(dois)/1000), time.time() - start_time))
    print('Total articles:',articles_df.shape[0])
    print('Final iteration:', iteration)
    print('Final time: %.2fs' % (time.time() - start_time))
```

```
    print()
    return articles_df


def fetch_threaded(dois: List[str], articles_df: pd.DataFrame, start_time: time.time, get_next: bool,
        runner_sleep: float, max_workers: int, max_articles: int) -> Tuple[List[str], pd.DataFrame]:
    next_dois = set()
    batch = 0
    while dois:
        # we get some DOIs
        dois_temp = dois[:1000]
        # request data for each DOI
        futures = runner(dois_temp, runner_sleep, max_workers)

        # once we're received all our DOI requests we can proceed
        # we create the articles dictionary for these 1000 articles
        # and add the corresponding reference DOIs to a list to be
        # parced in next iteration of caller
        articles = {}
        articles_togo = max_articles - articles_df.shape[0] - len(dois)
        queried_dois = set(articles_df.index).union(set(dois))
        for future in as_completed(futures):
            doi_temp, dict_temp = future.result()
            articles[doi_temp] = dict_temp
            # if we're not parsing the next ancestor level, let's save some time;
            # also, once we have enough total articles, let's not look for more ancestors
            if (get_next) and (len(next_dois) < articles_togo):
                next_dois = next_dois.union(parse_dois(dict_temp).difference(queried_dois))

        batch += 1
        # create corresponding DataFrame for articles and add to ongoing DataFrame
        articles_df_temp = articles_json_to_df(articles)
        articles_df = articles_df.append(articles_df_temp)
        print('** Duplicates dropped: %d **' % articles_df[articles_df.index.duplicated()].shape[0])
        articles_df = articles_df[~articles_df.index.duplicated(keep='
```

```python
first')]
        # articles_df.to_csv('backup.csv', index=True)
        print('** end of batch     #%d (%d articles) (%d batches next iteration) (%.2fs) **'
              % (batch, articles_df.shape[0], math.ceil(len(next_dois)/1000), time.time() - start_time))
        # remove the DOIs we just queried
        dois = dois[1000:]

    return list(next_dois), articles_df


def runner(doi_list: List[str], runner_sleep: float, max_workers: int) -> List:
    futures = []
    cr = Crossref(mailto = "your@email.ext")
    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        for doi in doi_list:
            futures.append(executor.submit(get_article_with_etiquette, doi, cr, runner_sleep))
            time.sleep(runner_sleep)
    return futures


def get_article_with_etiquette(doi: str, cr: Crossref, runner_sleep: float) -> Tuple[str, Dict]:
    count = 0
    while count < 100:
        try:
            r = cr.works(ids = doi)
        except Exception as e:
            error_substring = str(e).split(':')[1].strip() if len(str(e).split(':')) > 1 else str(e)
            if error_substring == 'Too Many Requests for url':
                extra_long = 300
                print('Sleeping for a while; %d seconds' % extra_long)
                time.sleep(extra_long)
            elif error_substring == 'Not Found for url':
                break
            elif error_substring in ['Max retries exceeded with url', 'Gateway Time-out for url']:
                print(error_substring, doi)
```

```
            else:
                print(e)
            time.sleep(runner_sleep)
            count += 1
        else:
            # bundle the article data with corresponding DOI, since `as_completed`
            # iterates based upon completion and not list order
            return doi, r.get('message')
    return doi, {}


# ARTICLE DATA TRANSFORMATION #

def get_key_keep_list() -> List:
    return ['DOI', 'author', 'reference']


def handle_prop_key(prop_key_str: str, prop_val):
    if prop_key_str == 'author':
        prop_list = list()
        # create a list, consisting of all
        # affiliations related to article
        for author in prop_val:
            affn_temp = author['affiliation']
            if len(affn_temp) > 0:
                prop_list.extend([x.get('name') for x in affn_temp if x.get('name')])
        if len(prop_list) > 0:
            return prop_list
        else:
            return None

    elif prop_key_str == 'reference':
        prop_list = list()
        # add all reference DOIs to a list
        for reference in prop_val:
            doi_temp = reference.get('DOI')
            if doi_temp:
                prop_list.append(doi_temp)
        if len(prop_list) > 0:
            return prop_list
```

```
        else:
            return None

    elif prop_key_str == 'published':
        date_list_nested = prop_val.get('date-parts')
        # return date in YYYY-MM-DD string format
        if date_list_nested:
            prop_val = ['-'.join([str(y) for y in x]) for x in date_list_nested]
        else:
            return date_list_listed

    # 'DOI', 'subject', 'title', 'container-title', 'reference-count' and non-null 'published' date
    # fall through to here
    if type(prop_val) == int:
        return prop_val
    elif len(prop_val) == 0:
        return None
    # if singleton list, just return the element
    elif isinstance(prop_val, list) and len(prop_val) == 1:
        return prop_val[0]
    else:
        return prop_val


def article_list_for_df(article_dict: Dict) -> List[dict]:
    key_keep_list = get_key_keep_list()
    article_list = list()

    # iterate through the articles
    for doi, prop_keys in article_dict.items():
        # iterate through the keys of the article
        if prop_keys:
            temp_dict = dict()
            for prop_key in prop_keys:
                if prop_key in key_keep_list:
                    temp_dict[prop_key] = handle_prop_key(prop_key, prop_keys[prop_key])
        else:
            temp_dict = {'DOI': doi}
        article_list.append(temp_dict)
```

```python

    return article_list


def articles_json_to_df(articles_json: json) -> pd.DataFrame:
    # dataframe creation
    articles_list = article_list_for_df(articles_json)
    articles_df = pd.DataFrame(data = articles_list, columns = get_key_keep_list())

    # some basic cleaning
    df_cols = articles_df.columns
    articles_df.fillna(np.nan, inplace=True)
    articles_df.set_index('DOI', inplace=True)
    articles_df = articles_df[~articles_df.index.duplicated(keep='first')]

    # basic variable creation and data type fixing
    articles_df['reference_len'] = articles_df['reference'].apply(lambda x: get_reference_len(x)).astype(int)
    articles_df['published'] = pd.to_datetime(articles_df['published'], errors = 'coerce')
    articles_df['year'] = articles_df['published'].dt.year
    articles_df['year'] = articles_df['year'].astype('Int64')
    articles_df.drop(['published'], axis=1, inplace=True)

    ## we want to know if we queried all references, so we need to keep all articles, ##
    ## even if they don't have author affiliations ##
    # primary affiliation to country mapping
    articles_df['country'] = articles_df['author'].apply(lambda x: get_primary_author_country(x))

    return articles_df


def get_primary_author_country(name_list: List[str]) -> str:
    if type(name_list) == list:
        return clean_affiliation_name(name_list[0])
    return np.nan


```

```python
def get_reference_len(reference: List) -> int:
    try:
        return len(reference)
    except TypeError:
        return 0


# ARTICLE PRIMARY AFFILIATION CLEANING #

def clean_affiliation_name(name: str) -> str:
    found = find_country_name_at_end_of_affiliation(name)
    if found:
        return found
    found = find_state_name_at_end_of_affiliation(name)
    if found:
        return found
    return None


def get_countries() -> Set[str]:
    with open('Countries.csv', 'r') as handle:
        countries = set([line.strip() for line in handle])
    return countries


def get_country_synonyms() -> Dict[str, str]:
    country_synonyms = {}
    with open('Country_synonyms.csv', 'r') as handle:
        for f in handle.readlines():
            k, v = f.split(',')
            country_synonyms[k.strip().lower()]=v.strip()
    return country_synonyms


def get_us_states() -> Set[str]:
    us_states = set()
    with open('US_states.csv', 'r') as handle:
        for f in handle.readlines():
            name, abbr = f.split(',')
            us_states.update([name.strip(), abbr.strip()])
    return us_states

```

```python

def find_country_name_at_end_of_affiliation(name: str) -> str:
    countries = get_countries()
    country_synonyms = get_country_synonyms()
    name = name.rstrip(' ,;-.()')
    country_regex = '|'.join(sorted(countries, key=len, reverse=True))
    match = re.search('(' + country_regex + ')$', name, re.I)
    if match:
        return match.group()
    country_synonyms_regex = '|'.join(sorted(country_synonyms.keys(), key=len, reverse=True))
    match = re.search('(' + country_synonyms_regex + ')$', name, re.I)
    if match:
        try:
            return country_synonyms[match.group().lower()]
        except KeyError:
            print('Synonym not found:', match.group().lower())
    return ''


def find_state_name_at_end_of_affiliation(name: str) -> str:
    us_states = get_us_states()
    name = name.rstrip(' ,;-.()')
    us_states_regex = '|'.join(us_states)
    zip_code_regex = '[0-9]{5}(-[0-9]{4})?'
    match = re.search('(' + us_states_regex + ')[, ]*' + zip_code_regex + '$', name, re.I)
    if match:
        return 'United States'
    return ''


# ARTICLE AGGREGATION #

def get_articles_from_csv(filename: str, dir='articles') -> pd.DataFrame:
    path = '/'.join([dir, filename])
    articles_df = pd.read_csv(path, index_col=0)
    if articles_df.empty:
        return articles_df

    # if needed, we need to rebuild the lists from string
```

```
representation
    columns = articles_df.columns
    if 'reference' in columns:
        articles_df['reference'] = articles_df['reference'].apply(lambda x: string_to_list(x))
    if 'author' in columns:
        articles_df['author'] = articles_df['author'].apply(lambda x: string_to_first_list_element(x))
    return articles_df


def string_to_list(str_of_list: str) -> list:
    if type(str_of_list) == str:
        return eval(str_of_list)
    else:
        return str_of_list


def string_to_first_list_element(str_of_list: str) -> str:
    if type(str_of_list) == str:
        return eval(str_of_list)[0]
    else:
        return str_of_list


def get_all_articles_from_csv() -> pd.DataFrame:
    articles_list = os.listdir('articles')
    articles_df = pd.DataFrame()
    for index, articles in enumerate(articles_list):
        print(index, articles)
        articles_temp_df = get_articles_from_csv(articles)
        articles_temp_df['country'] = articles_temp_df['country'].apply(lambda x: x.title() if type(x) == str else x)
        articles_df = articles_df.append(articles_temp_df)
        articles_df = articles_df[~articles_df.index.duplicated(keep='first')]
    return articles_df


def get_cnty_counts_df(articles_input_df: pd.DataFrame) -> pd.DataFrame:
    # we create a new derivative DataFrame
```

```python
    articles_df = articles_input_df.drop(['author'], axis=1)

    # default ratio value when no references present (refs == None)
    articles_df['local_ratio'] = np.nan
    # default count when no references present
    articles_df['reference_used'] = 0

    # we don't want to drop articles without a country,
    # but we only want to consider source articles with a country
    print(articles_df[~articles_df['country'].isna()].shape)
    for index in articles_df[~articles_df['country'].isna()].index:
        refs = articles_df.loc[index, 'reference']
        # refs should either be non-empty list or np.nan
        if type(refs) == list:
            source_cnty = articles_df.loc[index, 'country']
            refs_in_cnty, refs_total = get_cnty_counts_for_refs(refs, articles_df, source_cnty)

            # either: the article does not have all DOI ancestors in DF, not a valid observation;
            #  the article has no references with a country, does not need to be updated
            if refs_total == 0:
                continue

            ratio_temp = refs_in_cnty / refs_total
            articles_df.loc[index, 'local_ratio'] = ratio_temp
            articles_df.loc[index, 'reference_used'] = refs_total

    # although strictly not needed since np.na will be ignored for grouped statistics by default,
    # doing this will reduce any human error
    articles_df.dropna(subset=['local_ratio'], inplace=True)
    articles_df.drop(['reference'], axis=1, inplace=True)
    # data type fix from float
    articles_df['reference_used'] = articles_df['reference_used'].astype(int)
    return articles_df


def get_cnty_counts_for_refs(refs: List[str], articles_df: pd.DataFrame, source_cnty: str) -> Tuple[int, int]:
```

```
    refs_in_cnty = 0
    refs_total = 0
    for ref in refs:
        ref = urllib.parse.unquote(ref.strip())
        try:
            cnty_temp = articles_df.loc[ref, 'country']
        # reference not in DataFrame
        except KeyError:
            pass
        # if successful
        else:
            # when entry is not np.nan
            if cnty_temp == cnty_temp:
                refs_total += 1
            if cnty_temp == source_cnty:
                refs_in_cnty += 1
    return refs_in_cnty, refs_total


def create_grouped_dfs(articles_df: pd.DataFrame) -> pd.DataFrame:
    articles_cnty_df = articles_df.groupby(['country']).mean()
    return articles_cnty_df
```

Listing 2: Python Functions Developed For Data Collection, Cleaning, and Transformation

R Code

```
library(mapdata)
library(tidyverse)
library(sf)
library(tigris)
library(spData)
library(tmap)
library(cartogram)
library(isdas)
library(gridExtra)
library(plotly)
library(patchwork)
library(spdep)
library(GWmodel)
library(kableExtra)
library(spatialreg)
library(spgwr)


########### Retrieve and Prepare Data ###########

data('World')
articles <- read_csv('country_counts.csv')
colnames(articles) <- c('sovereignt', 'ref', 'local_ratio', 'reference_used')
articles$sovereignt[articles$sovereignt == 'United States'] <- 'United States of America'
articles$sovereignt[articles$sovereignt == 'Tanzania'] <- 'United Republic of Tanzania'
articles$sovereignt[articles$sovereignt == 'Serbia'] <- 'Republic of Serbia'
merged <- merge(World, articles, by='sovereignt')

drop_cnty <- c(
  "Hong Kong",
  "Singapore",
  "Puerto Rico",
  "Falkland Is.",
  "Fr. S. Antarctic Lands",
  "New Caledonia",
  "Greenland"
```

```
)
merged <- subset(merged, !(name %in% drop_cnty))

merged.sp <- as(merged, "Spatial")
merged.nb <- poly2nb(pl = merged.sp, queen = TRUE)

World_centroids <- suppressWarnings(
  st_centroid(merged) %>%st_coordinates()
)

########### Country Adjacency ###########

plot(merged.sp, border = "gray")
plot(merged.nb, coordinates(merged.sp), col = "red", add = TRUE)

World.dnb <- dnearneigh(World_centroids, d1 = 0, d2 = 15)
plot(merged.sp, border = "gray")
plot(World.dnb, World_centroids, col = "brown", add = TRUE)

World.dnb <- dnearneigh(World_centroids, d1 = 0, d2 = 25)
plot(merged.sp, border = "gray")
plot(World.dnb, World_centroids, col = "brown", add = TRUE)

World.knb <- knn2nb(knearneigh(World_centroids, k = 3))
plot(merged.sp, border = "gray")
plot(World.knb, World_centroids, col = "orange", add = TRUE)

World.knb <- knn2nb(knearneigh(World_centroids, k = 5))
plot(merged.sp, border = "gray")
plot(World.knb, World_centroids, col = "orange", add = TRUE)


########### Local Ratio Mapping ###########

local_ratio_map <- tm_shape(merged)+
    tm_fill(col = 'local_ratio', title = 'Local Ratio',
            breaks = c(0, 0.05, 0.1, 0.15, 0.2,0.3, 0.4, 0.6, 1),
            palette = 'viridis')+
    tm_layout(main.title = "Local Ratios Calculated per Country",
              main.title.position = "center")

local_ratio_map
```

```r


########### Local Ratio Analysis ###########

corr_list <- list()
for (m in list(3, 5, 10)) {
  # Find the m nearest neighbors based on the centroids of each country.
  World.knb <- knn2nb(knearneigh(World_centroids, k = m))

  # `country_sma` will hold the estimated local ratios, based on neighbors.
  country_sma <- list()

  # Iterate over countries.
  for (i in 1 : nrow(merged)) {
    neighbor_list <- World.knb[[i]]
    local_ratios_sum <- 0
    for (j in 1 : m) {
      local_ratio <- merged[neighbor_list[[j]],]$local_ratio
      local_ratios_sum <- local_ratios_sum + local_ratio
    }
    country_sma <- append(country_sma, local_ratios_sum/m)
  }

  # Create data frame for the current m value.
  country_sma <- as.numeric(country_sma)
  assign(paste("knb_df_", m, sep = ""), data_frame(merged$sovereignt, merged$local_ratio, country_sma))
}

# Rename columns.
colnames(knb_df_3) <- c('sovereignt','local_ratio', 'country_sma')
colnames(knb_df_5) <- c('sovereignt','local_ratio', 'country_sma')
colnames(knb_df_10) <- c('sovereignt','local_ratio', 'country_sma')

# Compute Pearson correlation coefficients.
cor(knb_df_3$local_ratio, knb_df_3$country_sma)
cor(knb_df_5$local_ratio, knb_df_5$country_sma)
cor(knb_df_10$local_ratio, knb_df_10$country_sma)

# Merge results into a single data frame that can be shown in the
```

```
article.
knb_df_selected <- data_frame(
  knb_df_10$sovereignt,
  knb_df_10$local_ratio,
  knb_df_3$country_sma,
  knb_df_5$country_sma,
  knb_df_10$country_sma)

colnames(knb_df_selected) <- c(
  'sovereignt',
  'Sample Local Ratio',
  '3 Neighbor Estimation',
  '5 Neighbor Estimation',
  '10 Neighbor Estimation')

# Create maps for the selected k-neighbors values.
empty_world <- World
world_with_estimates <- merge(empty_world, knb_df_selected , by = 'sovereignt', all=T)

three_neighbor_map <- tm_shape(world_with_estimates)+
  tm_fill(col = '3 Neighbor Estimation', title = '3 Neighbor Ratio Estimation',
          breaks = c(0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.6, 1),
          palette = 'viridis')
three_neighbor_map

five_neighbor_map <- tm_shape(world_with_estimates)+
  tm_fill(col = '5 Neighbor Estimation', title = '5 Neighbor Ratio Estimation',
          breaks = c(0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.6, 1),
          palette = 'viridis')
five_neighbor_map

ten_neighbor_map <- tm_shape(world_with_estimates)+
  tm_fill(col = '10 Neighbor Estimation', title = '10 Neighbor Ratio Estimation',
          breaks = c(0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.6, 1),
          palette = 'viridis')
ten_neighbor_map

# Compute Moran I test statistics
```

```r

# 3 neighbors: simple average
merged.knb_for_moran_3 <- knn2nb(knearneigh(World_centroids, k = 3))
merged.w3 <- nb2listw(merged.knb_for_moran_3)
moran.test(merged$local_ratio, merged.w3)

# 3 neighbors: IDW
merged.idw_w3 <- nb2listwdist(merged.knb_for_moran_3, merged)
moran.test(merged$local_ratio, merged.idw_w3)

# 5 neighbors: simple average
merged.knb_for_moran_5 <- knn2nb(knearneigh(World_centroids, k = 5))
merged.w5 <- nb2listw(merged.knb_for_moran_5)
moran.test(merged$local_ratio, merged.w5)

# 5 neighbors: IDW
merged.idw_w5 <- nb2listwdist(merged.knb_for_moran_5, merged)
moran.test(merged$local_ratio, merged.idw_w5)

# 10 neighbors: simple average
merged.knb_for_moran_10 <- knn2nb(knearneigh(World_centroids, k = 10))
merged.w10 <- nb2listw(merged.knb_for_moran_10)
moran.test(merged$local_ratio, merged.w10)

# 10 neighbors: IDW
merged.idw_w10 <- nb2listwdist(merged.knb_for_moran_10, merged)
moran.test(merged$local_ratio, merged.idw_w10)
```

Listing 3: R Visualization and Analysis of Local Ratios