National Centre for Geocomputation (NCG), Significant Developments in Geographic Information Science

Sean Kinch

Summer 2024

Abstract

The aim of my SPUR 2024 project was to catalogue the books in the office of the late Dr. Martin Charlton in an effort to preserve, spread, and memorialize his lifetime of work. I then go on to study the data and trends in the study of Geocomputation through the lens of the bookshelf. The catalog is available here or through my SPUR website here.

Introduction

Since its inception in 2004 with the support of Science Foundation Ireland and centred in Maynooth University, the NCG, has become firmly established as a leading centre for research in the field of Geocomputation, applying computational methods to large spatial data sets from acquisition to analysis, modelling and visualisation. As a leading research centre the NCG is committed to:

- Extending the understanding and utilisation of spatial data from postgraduate programmes to national and international research project engagement
- Applying existing geotechnologies to specific problems for enhanced decision making
- new innovative technologies and applications for real-time response to real world challenges [1]

Dr. Martin Charlton played an important role in the National Centre for Geocomputation as a co-founder and important member of the team as highlighted in these texts by Prof. Chris Brunsdon and Maynooth University. He left behind an office of crucial and useful material spanning the beginnings of Geocomputation and long before. The importance of cataloging this library is to create a convenient way for students and researchers to reach and use this wealth of material in a productive manner to forward the field of study through papers, discussion etc.

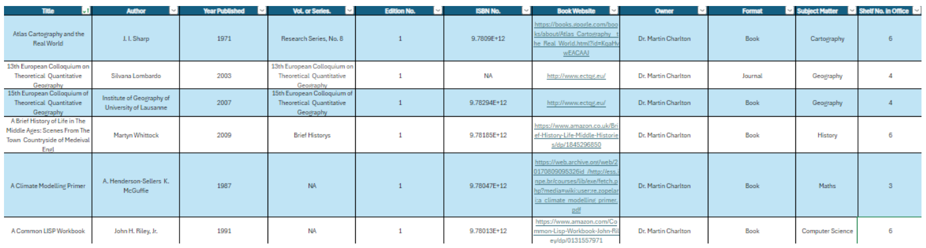
Catalogue

With well over 1000 books, Excel was the ideal tool for this task. The factors in choosing this software came down to ease of use, collaboration and features in organisation, validation, and querying. Formatting in CSV also has benefits in many languages.

I used any and all fields of relevance such as Title, Year Published, ISBN , Series, Edition, etc. in the catalog to provide a comprehensive, detailed report of the library. All of these fields give a huge data set ripe to analyse and interpret.These fields were populated with the data of the books taken off the shelf. The books ranged from journals of geographers in the early 1900's to papers of cutting edge methods and techniques from today.

I used a blend of manual entry and web scraping (Python script) shown here,

```python
import requests
from openpyxl import Workbook

def get_book_details(isbn):
    # Google Books URL
    url = f"https://www.googleapis.com/books/v1/volumes?q=isbn:{isbn}"

    response = requests.get(url)

    if response.status_code != 200:
        return None

    data = response.json()

    if 'items' not in data:
        return None

    book_info = data['items'][0]['volumeInfo']

    title = book_info.get('title', 'N/A')
    authors = ', '.join(book_info.get('authors', ['N/A']))

    return {
        'ISBN': isbn,
        'Title': title,
        'Author': authors
    }

def SaveExcel(book_details, filename='SPURCatalogue.xlsx'):
    workbook = Workbook()
    sheet = workbook.active

    headers = ['ISBN', 'Title', 'Author'] #Fields
    sheet.append(headers)

    for book in book_details:
        sheet.append([book['ISBN'], book['Title'], book['Author']])

    workbook.save(filename)

def main():
    isbns = [
        '9780471985464'  # Example
    ]

    book_details = []

    for isbn in isbns:
        details = get_book_details(isbn)
        if details:
            book_details.append(details)

    SaveExcel(book_details)

if __name__ == "__main__":
    main()
```



Title	Author	Year Published	Vol. or Series.	Edition No.	ISBN No.	Book Website	Owner	Format	Subject Matter	Shelf No. in Office
Atlas Cartography and the Real World	J. I. Sharp	1971	Research Series, No. 8	1	9.7809E+12	https://books.google.com/books/about/Atlas_Cartography_the_Real_World.html?id=KqaHvwEACAAJ	Dr. Martin Charlton	Book	Cartography	6
13th European Colloquium on Theoretical Quantitative Geography	Silvana Lombardo	2003	13th European Colloquium on Theoretical Quantitative Geography	1	NA	http://www.ectqg.eu/	Dr. Martin Charlton	Journal	Geography	4
15th European Colloquium of Theoretical Quantitative Geography	Institute of Geography of University of Lausanne	2007	15th European Colloquium of Theoretical Quantitative Geography	1	9.78294E+12	http://www.ectqg.eu/	Dr. Martin Charlton	Book	Geography	4
A Brief History of Life in The Middle Ages: Scenes From The Town Countryside of Medeival Engl	Martyn Whittock	2009	Brief Historys	1	9.78185E+12	https://www.amazon.co.uk/Brief-History-Life-Middle-Histories/dp/1845296850	Dr. Martin Charlton	Book	History	6
A Climate Modelling Primer	A. Henderson-Sellers K. McGuffie	1987	NA	1	9.78047E+12	https://web.archive.org/web/20170809095326id_/http://ess.inpe.br/courses/lib/exe/fetch.php?media=wiki:user:re.zopelar:ica_climate_modelling_primer.pdf	Dr. Martin Charlton	Book	Maths	3
A Common LISP Workbook	John H. Riley, Jr.	1991	NA	1	9.78013E+12	https://www.amazon.com/Common-Lisp-Workbook-John-Riley/dp/0131557971	Dr. Martin Charlton	Book	Computer Science	6

Figure 1: Excel Catalogue

This script generally worked well with books published post 1970 when the ISBN became standard practice. These entries were checked to ensure accurate and complete information. Many of these books and journals came before this time and were so compensated with links to the society which published their respective journal and other useful information. Isbnlib is a Python library which was used heavily in this project that provides several useful methods and functions to validate, clean, transform, hyphenate and get metadata for ISBN strings. For handling special cases such as books, journals, papers with no

ISBN, entering ”NA” or ”NaN” into the cells allows Python’s panda library to handle these cases effectively. Group-o ISBN publisher codes, ex. 0086297466, were handled cleanly by the Excel software with the use of specific formats for the entries. However if the leading zeros were dropped in the process,a python script can format these ISBN-10s correctly,

```
def add_leading_zeros(isbn):

    isbn = isbn.strip()

    if len(isbn) < 10:
        isbn = isbn.zfill(10)

    return isbn

def process_isbn_list(isbn_list):

    return [add_leading_zeros(isbn) for isbn in isbn_list]

def save_to_excel(isbn_list, filename='ISBN10.xlsx'):
    from openpyxl import Workbook

    workbook = Workbook()
    sheet = workbook.active

    sheet.append(['ISBN-10'])

    for isbn in isbn_list:
        sheet.append([isbn])

    workbook.save(filename)

if __name__ == "__main__":
    isbn_list = [
        "123456789"  #Example
    ]

    processed_isbns = process_isbn_list(isbn_list)

    save_to_excel(processed_isbns)
```

Data Analysis

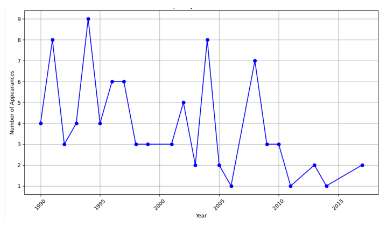
My main targets in this data set were to analyse a number of avenues of inquiry such as the geographic data of publishing locations, the network of coauthors, and the appearance of selected subjects over time periods. These tasks were done over a number of useful software's such as Gephi, Excel and Python scripts.

Trends

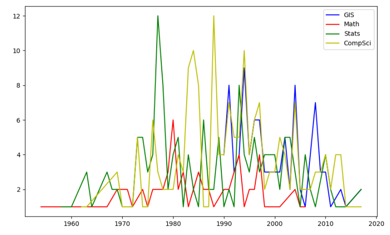
To view trends in subjects such as GIS, Computer Science, etc. over the course of the bookshelf, I wrote a Python script to plot selected subjects as line charts over a period from 1970-2020 as this is the ISBN age of books and captures the latest entry's in the office as Dr. Martin Charlton retired from his role in the NCG in 2020. This python script here,

```
import pandas as pd
import matplotlib.pyplot as plt

input_csv_path = r'C:Catalogue.csv'
df = pd.read_csv(input_csv_path)

df['GIS_Appearance'] = df['Subject Matter'].str.contains('GIS', case=False, na=F
#Add

gis_counts_by_year = df[df['GIS_Appearance']].groupby('Year').size()
#Add

plt.figure(figsize=(10, 6))
plt.plot(gis_counts_by_year.index, gis_counts_by_year.values, label='GIS',
linestyle='-', color='b')
#Add

plt.legend()
plt.show()
plt.title('Subject Matter vs Time')
plt.xlabel('Year')
plt.ylabel('Number of Appearances')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.xlim([1960, 2020])

plt.show()
```

Which results in the following plots,



(a) GIS vs Time



(b) Subject vs Time

Trend Graphs

The graph (a) on the left shows the appearance of books on GIS over the years. We see the first appearance of these books begins at the start of the 1990's which is backed by this paper, Twenty years of progress: GIScience in 2010 by Michael F. Goodchild[2]. A big takeaway from this graph is the overall decline in the publications of books and journals written on GIS as a broad stroke. Due to the advancement of the field reasearchers and academics are now honing in on fruitful areas within GIS and so has given rise in the publication of subjects such as remote sensing, spatial analysis, etc. The graph (b) on the right illustrates the underpinning of core subjects such as math, statistics, and computer science which allow the sophisticated softwares such as ARC and others to exist. These books will have more publications as they allow the techniques and methods within GIS to exist.

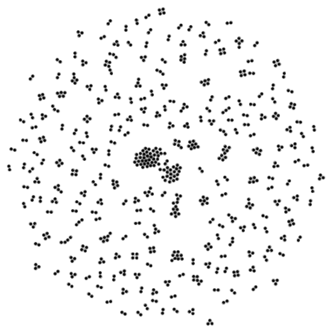
I conjecture that from this data set we can infer that the larger, global trends follow from this.

Network

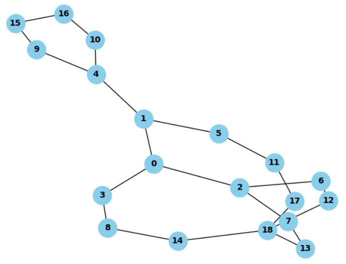
To prepare the data for Gephi, which is the software I worked with for this network visualisation, I used a Python script to iterate through the Author's and create a coauthor CSV file. Python script,

```
import pandas as pd
from itertools import combinations

df = pd.read_csv(r'C:Catalogue.csv', encoding='ISO-8859-1')

authors_df = df[['Title', 'Author']]

def extractcoauthor(authors):
    authors_list = authors.split(', ')
    pairs = list(combinations(authors_list, 2))
    return pairs

coauthor = []

for _, row in authors_df.iterrows():
    authors = row['Author']
    if pd.notna(authors):
        pairs = coauthor(authors)
        coauthor.extend(pairs)

coauthor_df = pd.DataFrame(coauthor, columns=['Source', 'Target'])

OutputFile = 'coauthor.csv'
coauthor_pairs_df.to_csv(OutputFile, index=False)

print(f'Co-author pairs saved to {OutputFile}')
```

These pairs gave me a source and target to construct the graph. I used ForceAtlas 2 layout algorithm as it gives an intuitive feel for the graph with it's central clustering, competence with large networks, and shows weighted nodes and edges effectively.



(a) Graph G



(b) Largest Component of G, g

Networks

The image (a) of Graph G is the entire coautorship network of the library. This network mimics any real world field of study with a dense core with many of the top contributors working closely together on the main areas of study with smaller clusters of few authors working together on more niche topics within the field[3]. The library cover's a huge spread of content over many disciplines resulting in a lot of sub groups within the network. Using Gephi's statistical analysis of the graph, it shows the average degree of G is 1.893 supporting our claim of many small clusters. The graph has a network diameter of 5 which indicates within the connected component of the graph nodes are highly interlinked. The graph is divided into 259 subgraphs and has a density of 0.03 meaning only 3% of all possible edged exist within the graph. A strong statistic to show collaborative groups in this network is a cluster coefficient given by the global clustering for an undirected graph,

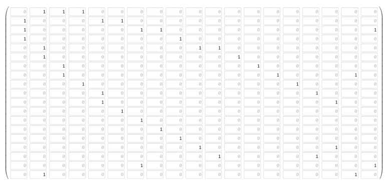
$$C = \frac{\sum\limits_{i,j,k} A_{ij}A_{jk}A_{ki}}{\frac{1}{2}\sum\limits_{i} k_i(k_i-1)}$$

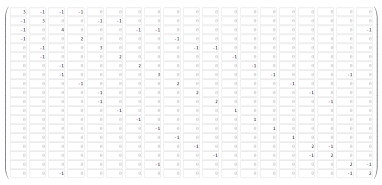
where:

$$k_i = \sum_j A_{ij}$$

which gives a value of 0.784[4].

The image (b) of a subgraph of G, g with matrices,


(a) Adjacency Matrix


(b) Laplacian Matrix

Matrices to subgraph of G

These graphs are found following from their definitons,

The adjacency matrix A of a graph G with n vertices is an $n \times n$ matrix where each element a_{ij} is defined as:

$$a_{ij} = \begin{cases} 1 & \text{if there is an edge between vertices } i \text{ and } j, \\ 0 & \text{otherwise.} \end{cases}$$

The Laplacian matrix L of a graph G with n vertices is defined as:

$$L = D - A$$

where D is the degree matrix and A is the adjacency matrix. Element-wise, the Laplacian matrix L is defined as:

$$l_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j, \\ -1 & \text{if there is an edge between vertices } i \text{ and } j, \\ 0 & \text{otherwise.} \end{cases}$$

These matrices come from the Vertex set V(g) and the Edge set E(g),

$$V(g) = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}$$

$$E(g) = \{\{0, 1\}, \{0, 2\}, \{0, 3\}, \{1, 4\}, \{1, 5\}, \{2, 6\}, \{2, 7\}, \{2, 18\}, \{3, 8\}, \{4, 9\}, \{4, 10\}, \{5, 11\}, \{6, 12\}, \{7, 13\}, \{7, 18\}, \{8, 14\}, \{9, 15\}, \{10, 16\}, \{11, 17\}, \{12, 13\}, \{13, 14\}, \{14, 15\}, \{15, 16\}, \{16, 17\}, \{17, 18\}\}$$

From these matrices we can use techniques from spectral graph theory to learn about the graph. Spectral graph theory is the study of the properties of a graph in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of matrices associated with the graph. These matrices and eigenvalues can be found using a python script for convenience,

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt

g = nx.Graph()

g.add_nodes_from(range(19))

edges = [#add]

g.add_edges_from(edges)

adjacency_matrix = nx.to_numpy_array(g)
print("Adjacency Matrix:\n", adjacency_matrix)

laplacian_matrix = nx.laplacian_matrix(g).toarray()
print("Laplacian Matrix:\n", laplacian_matrix)

eigenvalues1 = np.linalg.eigvals(adjacency_matrix)
print("Eigenvalues of the Adjacency matrix are:", eigenvalues1)

eigenvalues2 = np.linalg.eigvals(laplacian_matrix)
print("Eigenvalues of the Laplacian matrix are:", eigenvalues2)
```

The eigenvalues for these matrices are as follows, Eigenvalues of the Adjacency matrix:

$$\begin{aligned}\lambda_A = [&2.53199806, -2.52649791, 2.19249371, -2.08191736, 1.75262126, -1.2996\\ &-0.82572888, -0.63130223, 1.44500621, 1.20702624, -0.11529323, 0.8584\\ &-1.61803399, -1.61803399, 1.24239936 \times 10^{-16}, 0.61803399, 0.61803399]\end{aligned}$$

Eigenvalues of the Laplacian matrix:

$$\begin{aligned}\lambda_L = [&5.66715706, 4.80937168, 3.97657859, 3.42984359, 3.31851419, 2.69537752,\\ &1.05120128, 0.87593742, 5.43073691 \times 10^{-16}, 0.08605304, 0.16520329, 0.27\\ &1.47629064, 3.61803399, 2.61803399, 0.38196601, 1.38196601]\end{aligned}$$

For the adjacency matrix, many features of the graph are observed in the eigen values such as,

Lemma: If G is bipartite, and λ is an eigenvalue of the adjacency matrix A, then so is $-\lambda$.

This does not apply to our graph as,

$$\forall - \lambda \nexists \lambda.$$

Meaning the graph is not bipartiate[5]. The reality of this shows that even within this Graph g, there contain strong subgroups even within working closely together with bridges for perhaps once off communication with outside authors. An eigenvalue close to 0 as is present in the above set means the presence of an isolated vertex in the graph. The graph is connected bwo the Perron-Frobenius Theorem stating,

For connected graphs, the spectral radius of the adjacency matrix is positive, and the corresponding eigenvector has all positive entries.

The spectral radius of this graph is 2.53 corresponding to its largest eigenvalue.

A property taken from the Laplacian matrix of graph g, is the Fiedler value. Given by the definition[6],

Defn: The Fiedler Value or the algebraic connectivity of a graph is the second smallest eigenvalue of its Laplacian matrix L.

With the second smallest eigenvalue in the set being 0.086 it shows a small cut-set which would disconnect an important part of the graph. This Fiedler value is an important part of Cheegers inequality.

The truth of this graph is that the nodes represent authors publishing on GIS (given below). Some of these edges represent books such as Geographic Information Science Systems by Paul A. Longley, Michael F. Goodchild, David J. Maguire, David W. Rhind written in 2001.

Map

An idea with the data was to create a map of publishing locations around the world to show where the main source of content was coming from, the ISBNS's had to be prepared in a csv file to allow further analysis to be done as this is where the information on the book is encoded. This is the python script for that,

```python
import isbnlib
import pandas as pd
import numpy as np
import time
from isbnlib.dev._exceptions import ISBNNotConsistentError
from urllib.error import URLError, HTTPError

def analyze_isbn(isbn, retries=3, delay=2):
    isbn = str(isbn)
    isbn = isbnlib.canonical(isbn)

    if not isbnlib.is_isbn10(isbn) and not isbnlib.is_isbn13(isbn):
        return {"ISBN": isbn, "Error": "Invalid ISBN"}

    for attempt in range(retries):
        try:
            # isbnlib >> wiki
            info = isbnlib.meta(isbn)
            if not info:
                return {"ISBN": isbn, "Error": "No information available"}

            isbn_info = {
                "ISBN": isbn,
                "Title": info.get('Title', 'Unknown'),
                "Authors": ", ".join(info.get('Authors', [])) if 'Authors' in in
                "Publisher": info.get('Publisher', 'Unknown'),
                "Year": info.get('Year', 'Unknown'),
            "Language": isbnlib.desc(isbn)
        }
        return isbn_info

    # Debugging

input_csv_path = r'C:ISBN.csv'
df = pd.read_csv(input_csv_path)

print("Columns in DataFrame:", df.columns)

if 'ISBN No.' in df.columns:
    isbn_column = 'ISBN No.'
else:
    isbn_column = df.columns[0]

df[isbn_column] = df[isbn_column].replace('NA', np.nan)
df[isbn_column] = df[isbn_column].dropna()
df[isbn_column] = df[isbn_column].apply(lambda x: f"{int(float(x))}" if isinstan

results = [analyze_isbn(isbn) for isbn in df[isbn_column]]

results_df = pd.DataFrame(results)

output_csv_path = r'C:ISBN.csv'
results_df.to_csv(output_csv_path, index=False)

print(f"Results saved to {output_csv_path}.")
```

This data was then ready to be visualised as a map. I used a technique from statistics, Kernel Density Estimation, to overlay a heatmap to emphasise the hotspots in publishing across the world. The Kernel Density Estimation is defined as,

Let $(x_1, x_2, \dots, x_n)$ be independent and identically distributed samples drawn from some univariate distribution with an unknown density f at any given point x. We are interested in estimating the shape of this function f. Its *kernel density estimator* is

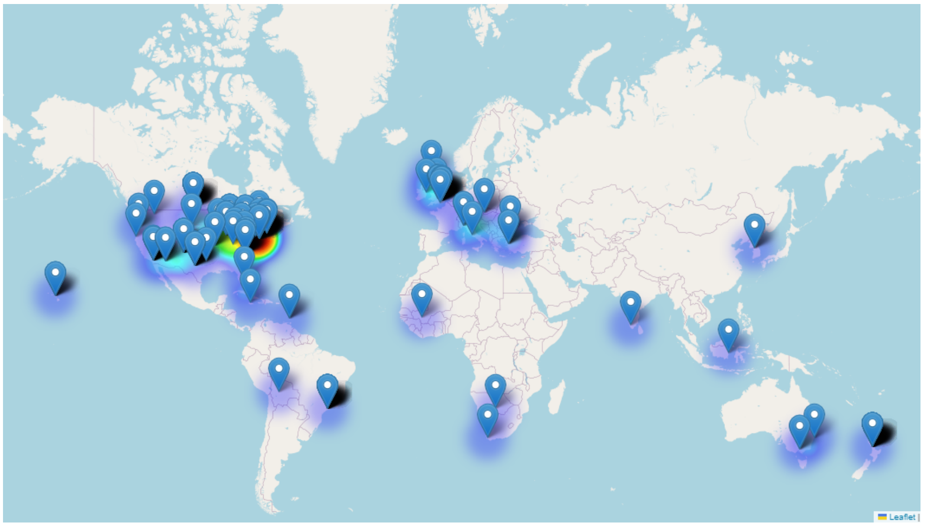
$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

where K is the *kernel*—a non-negative function—and $h > 0$ is a smoothing parameter called the *bandwidth* or simply width. A kernel with subscript h is called the *scaled kernel* and is defined as $K_h(x) = \frac{1}{h}K\left(\frac{x}{h}\right)$.

This process was done with a Python script making use of its library to handle the work,

```
import geopandas as gpd
from geopy.exc import GeocoderUnavailable, GeocoderTimedOut
import pandas as pd
import time
import folium
from sklearn.neighbors import KernelDensity
import numpy as np
from folium.plugins import HeatMap

def geocode_publisher_country(df, country_column, retries=3, delay=5, timeout=2
    if country_column not in df.columns:
        raise KeyError(f"Column '{country_column}' not found in the DataFrame.")

    df = df.dropna(subset=[country_column])
    unique_countries = df[country_column].unique()
    print("Geocoding the following countries:")
    print(unique_countries)

    for attempt in range(retries):
        try:
            gdf = gpd.tools.geocode(unique_countries, provider='nominatim', user
            gdf[country_column] = unique_countries
            return gdf
        except (GeocoderUnavailable, GeocoderTimedOut) as e:
            print(f"Attempt {attempt + 1} failed: {e}")
            if attempt < retries - 1:
                time.sleep(delay)
            else:
                print("Max retries exceeded. Geocoding failed.")
                return None
        except Exception as e:
            print(f"Unexpected error during geocoding: {e}")
            return None

def main(input_csv, country_column):
    df = pd.read_csv(input_csv)
    print(df.head())

    if country_column not in df.columns or df[country_column].isna().all():
        print(f"Error: '{country_column}' column is missing or contains only NaN
        return None

    df_geocoded = geocode_publisher_country(df, country_column)

    if df_geocoded is not None:
        df_merged = pd.merge(df, df_geocoded, on=country_column, how='left')

        df_merged['Latitude'] = df_merged['geometry'].apply(lambda p: p.y if p e
        df_merged['Longitude'] = df_merged['geometry'].apply(lambda p: p.x if p

        return df_merged
    else:
        print("Geographical analysis failed due to geocoding service issues.")
        return None

input_csv = r'C:ISBN.csv'
country_column = 'Publisher'
output_html = r'C:GeographicalAnalysis.html'

df_geocoded = main(input_csv, country_column)

if df_geocoded is not None:
    df_geocoded.to_html(output_html, index=False)
    print(f"Results saved to {output_html}.")

    m = folium.Map(location=[20, 0], zoom_start=2)

    latitudes = df_geocoded['Latitude'].dropna()
    longitudes = df_geocoded['Longitude'].dropna()

    if len(latitudes) > 0 and len(longitudes) > 0:
        xy = np.vstack([longitudes, latitudes]).T

        kde = KernelDensity(bandwidth=1.0, kernel='gaussian')
        kde.fit(xy)

        z = np.exp(kde.score_samples(xy))

        z = (z - z.min()) / (z.max() - z.min())
        heat_data = np.vstack([latitudes, longitudes, z]).T

        HeatMap(heat_data, min_opacity=0.3, max_zoom=18).add_to(m)

    for _, row in df_geocoded.iterrows():
        if not pd.isna(row['Latitude']) and not pd.isna(row['Longitude']):
            folium.Marker(
                location=[row['Latitude'], row['Longitude']],
                popup=row[country_column],
            ).add_to(m)

    map_output_path = r'C:GeographicalMap.html'
    m.save(map_output_path)
    print(f"Map saved to {map_output_path}.")
else:
    print("Map creation skipped due to geocoding issues.")
```

This is the result of these scripts,



Map of Publishing Locations

From the heat map its clear that America is the dominant publisher on these STEM based books such as math, computer science, and GIS. This is a logical conclusion even without the heat map as the term GIS was coined and really grew from symposiums in univesitys such as University of Washington, University of Santa Barbara, etc [2]. The second highest density is based in the UK as prestigious colleges such as Cambridge and Oxford have well known, reputable publishing's and the science has spread over time while continuing to grow. Another reason for the UK heavy base is that Dr. Martin Charlton had a post in the University of Newcastle upon Tyne and brought many local books from his time there.

Advancements in Geocomputation

Geographically Weighted Regression is a good example as it's a technique developed by Charlton along with Brunsdon and Fotheringham. GWR is a local version of spatial regression that generates parameters disaggregated by the spatial units of analysis and given by the expression[8],

Let $\mathbf{y}$ be the dependent variable, and $\mathbf{X}$ be the matrix of independent variables, with each observation i associated with coordinates (u_i, v_i). The GWR model can be expressed as:

$$y_i = \beta_0(u_i, v_i) + \sum_{k=1}^{p} \beta_k(u_i, v_i) x_{ik} + \epsilon_i$$

where:

- y_i is the dependent variable at location i.
- x_{ik} is the k-th independent variable at location i.
- $\beta_k(u_i, v_i)$ are the location-specific regression coefficients that vary across space.
- ϵ_i is the error term at location i.

The coefficients $\beta_k(u_i, v_i)$ are estimated using weighted least squares, where the weights decrease with the distance between locations i and j. The weight for location j relative to location i is given by a kernel function $K(d_{ij})$, where d_{ij} is the distance between i and j:

$$W_i = \text{diag}(K(d_{i1}), K(d_{i2}), \ldots, K(d_{in}))$$

This technique started in the 90's and was computationally heavy due to the process and machines of the time. Working on large data sets was a difficult task and so the technique had limitations and wasnt practical for many scenarios. However with modern computing resources, GWR is a much more efficient process. With a collaborative effort, new models of GWR are being developed showing great benefitsuch as this paper on ,Generalized Geographically Weighted Regression Model within a Modularized Bayesian Framework[9]. The techinque as a whole is perfectly capable of handling much larger data sets covering a much greater field while executing in shorter time. The time complexity has been optimized greatly. Many of the top softwares available such as ArcGIS feature a GWR tool making it accessible to many researchers. With the ever advancing capabilities of computers, such as quantum computing, this trend will only continue.

This is important as GWR is widely used in many impactful subjects we have as a society such as urban planning and enviromental science. We rely on this technology to grow efficiently as a people and many other techniques in GIS such as path finding or kriging with many tangible benefits also show these same results.

Conclusion

Summary

To summarize, there is now an online catalogue available here in attempt to spread the material to researchers and students, this data had many interesting conclusions reached, and the field has advanced enormously and continues to do so to the benefit of everyone.

Future Plans

I would be pleased if more libraries, offices, etc. joined in or integrated this catalogue as it only helps to reach hard to find books.

There are many topics to keep an eye on in the advance of the field such as optimization of algorithms, I have an interest in the use of group theory to speed up heuristic algorithms [10] or Randomized Linear Algebra [11]

Refererences

1) Maynooth University
2) Michael F. Goodchild
3) National Library of Medicine
4) Matthieu Latapy, Main-memory Triangle Computations for Very Large (Sparse (Power-Law)) Graphs, in Theoretical Computer Science (TCS) 407 (1-3), pages 458-473, 2008
5)David P. Williamson
6) Justin Wyss-Gallifent
7) Gerard Rushton, Michael F. Goodchild, Larence M. Ostresh Jr., Nicholas J. Cox, David W. Rhind, David J. Maguire, B. William Hickin, Alan J. Strachan, Bradley O. Parks, Louis S. Steyaert, Paul A. Longley , Sue M. Brooks, Rachael McDonnell, Bill MacMillan, Thomas J. Cova, Harvey J. Miller, Kate Beard, Andrew U. Frank, and Jiawei Han
8)Geographically Weighted Regression
9) Yang Liu and Robert J. B. Goudie
10) Yichao We and Xizhao Wang
11) arXiv Paper