PEERAGOGY AND TEACHING

M. Tedeschi[1], S. Belich[2], P. Ricaurte Quijano[3], C. Danoff[4], J. Corneli[5], S. Ayloo[2]

[1]Pace University (UNITED STATES)
[2]New York College of Technology (UNITED STATES)
[3]Technologico de Monterrey (MEXICO)
[4]Mr. Danoff's Teaching Laboratory (UNITED STATES)
[5]Oxford Brookes University (UNITED KINGDOM)

Abstract

We are part of a team of collaborators from across the globe working with synchronous and asynchronous technologies to collaboratively explore and develop 'Peeragogy', a collection of practical techniques for collaborative learning and collaborative work. Use cases for these techniques include peer learning and research co-production. Alongside our shared informal learning in the Peeragogy Project, several of us are employed as instructors in formal education settings. Here, we explore the interface between these two worlds. This paper shows how a graduate class was instructed to use Beautiful Soup for a final project in a web assisted course. We started with step-by-step installation instructions for both Windows and Unix systems, and quickly moved into practical applications. The focus was on retrieving data from major educational and economic websites, such as those of the World Bank. We used standard teaching methods to explain the tools and assignments. Students then worked together in groups, teaching each other in a more peeragogical manner. The results of these student collaborations will be described. In addition, we reflect briefly on other examples at the interface between peeragogy and formal teaching. Our conclusions point to the need for more work on refining techniques that can support peeragogy within traditional educational settings.

Keywords: AI, education, paragogy, peeragogy, peer learning.

1 INTRODUCTION

The education system is undergoing dynamic and volatile change. Educators are increasingly likely to encounter a high prevalence of remote students, and a growing use of Artificial Intelligence (AI). Meanwhile, students and teachers alike can avail themselves of a plethora of learning resources, including free videos, commercial course providers, and various Open Educational Resources (OER). Teaching environments range from custom 'hyflex' classrooms [1] to classrooms with no computers where students continue to get value from printed materials. Technology enhanced learning shows promise, but such applications are not without contention [2].

These dynamic changes and variability in the learning environment can be challenging for pedagogues, but they can also create new *peeragogical* possibilities. Peeragogy is the word we use to describe a growing collection of practical techniques for collaborative learning and collaborative work. This paper looks at how peeragogy can complement pedagogical practice, to develop high-quality learning experiences.

The authors of this paper have been exploring peeragogy together, using various synchronous and asynchronous technologies, as part of the Peeragogy Project, which has been running since 2012. Examples of our joint output include the Peeragogy Handbook published on peeragogy.org, as well as research papers including "Patterns of Peeragogy" [3]. We also get value from regular informal interactions, sharing feedback across our different spheres of professional endeavours. In particular, working together across domains and geographies, we continue to develop our individual approaches to teaching, learning, facilitation, project management, and more.

Teaching teachers to teach effectively, and teaching students to learn effectively are foundational tenets of mainstream education. Teachers often strive to keep classrooms active, because active learning is particularly effective. 'Pair learning' is one of the common approaches to structuring activities, in which each student works together with another student. This way, they develop communication skills alongside new subject knowledge. It can motivate the students to learn and participate in learning. Both peeragogical and pedagogical learning designs can make use of the same underlying theories of education, such as Vygotsky's Zone of Proximal Development [4].

Here, we will focus on a particular learning scenario that was used in one of our classrooms. This case study explores the experience of Python and web-scraping techniques at the university level. The class started with step-by-step installation instructions for both Windows and Unix systems, and quickly moved into practical applications. The focus was on retrieving data from major educational and economic websites, such as the World Bank. Students learned how to parse HTML, extract relevant information, and pre-process the data for use in charting and analytical tools. By the end, they had a solid grasp on creating insightful visualizations and aiding in the exploration of current trends and statistics in the educational field. This hands-on approach ensured students left with the skills to scrape and analyze web data effectively.

In particular, we used methods inspired by Peeragogy to teach the students, and, later, to reflect on that work. Students worked together in groups, teaching each other. The students' collaborative results will be described in the paper. We round out our discussion of this case with a brief look at other examples of peeragogical ideas being used in teaching and learning activities: in a high school English classroom, in an English as a Second Language (ESL) class for adults, in an informal online learning group, and in both assessed and extracurricular data science projects for university students.

2 METHODOLOGY

In the Peeragogy project, we meet on a regular basis to co-create and co-learn. We document our learning together, share our ideas and generate new ideas. We have made particular use of the Design Patterns tradition ([3], [5], [6]) to create reusable but suitably flexible toolkits for structuring learning activities. Broadly, we explore technologies, methods, and experiences together, and seek to make them relevant to others. Examples include collaborative design workshops and other structured peer learning experiences. Following both in-house and public-facing activities, we analyze what went well, lessons learned and what can be improved. This methodology of shared reflection brackets the specific exercise described below.

2.1 Beautiful Soup

One of our learning initiatives is teaching students python and web-scraping techniques. Web scraping has become an essential technique for extracting valuable information from the vast amount of unstructured data available on the internet. *Beautiful Soup* [7] is a python [8] library that can be used for web scraping. It provides tools for pulling data out of HTML and XML files. With Beautiful Soup, you can easily navigate through the document structure, search for specific elements, and extract the desired information.

It is particularly useful when you need to scrape data from web pages for various purposes like data analysis, data visualization (via pandas [9] and matplotlib [10]), research, or automation tasks. The final project is a program which is assumed to be executed as a daily script. In addition to the requirements listed in Fig. 1 (below), students were also asked to consider repeatability and reproducibility of their results. To improve repeatability, the student could: 1) add robust error handling to handle unexpected situations gracefully, 2) ensure that the code is flexible enough to handle variations in website structures or data formats, 3) write modular and reusable code that can be easily adapted for different websites or use cases, and 4) document the code thoroughly to explain its purpose, assumptions, and limitations

Students were encouraged to refactor their code with these considerations in mind. Students worked in teams to complete the project: our class had 6 teams of 3 students each.

Basic requirements

- It must pull data from the internet (typically webpages)
- It must pull data from 5 different webpages (using pandas)
- It must do some basic calculations on numerical data from 3 of those pages (mean, median, mode, min, max)
- It must make use of your own methods where appropriate
- It must produce at least 3 charts from 3 of those pages
- It must save the data to a file

Additional requirements (must have 4 of the following, more than 4 is extra credit):

- 1 of the webpages includes data which changes daily
- The 5 webpages are from 5 different websites
- It does basic calculations on all 5 of those webpages
- It uses Beautiful Soup on 1 of the webpages and parses the html
- It produces 5 charts from all 5 of the webpages
- It saves the charts in addition to the data, to image files
- It reads the previous file, and notes if any webpage has changed data* (hard)

Figure 1: Basic and additional requirements for student projects in the Beautiful Soup exercise.

3 RESULTS

All students were able to finish the project, at the level of meeting the basic requirements, and producing a demo. The instructor, however, was unable to reproduce all their results, which was a key part of the assignment.

Many students had little to no background as python programmers (e.g., one was a C++ engineer). Many had difficulty taking a verbal description and producing an algorithm. Their approaches ended up being quite varied, for instance, one group chose to use Selenium [11] instead of Beautiful Soup, which at first seemed to be the wrong choice. The group was allowed to proceed.

Only two of the teams' work was reproducible in entirety. They used different environments (Jupyter [12] and Visual Studio [13]). Without standardization, the process ended up being a little chaotic. The teams were a mix of weak and strong programmers. Each team was able to present a project, however. They were allowed multiple attempts.

One group used a movie website and only 2 websites were running when trying to reproduce the code. The assignment had asked them to pull data from 5 different web pages (using pandas), and this was not really accomplished in a reproducible manner. Some groups had 2 or 3 web pages that worked, though, again, their calculations were not reproducible in entirety. A sample of working code is given in our appendix together with some illustrative student findings. The code was retested and verified.

3.1 Results from the World Bank Dataset

Table 1. Sample of Scraped Data

	Country	***Population***
0	India	1,450,935,791
1	China	1,419,321,278
2	United States	345,426,571
3	Indonesia	283,487,931
4	Pakistan	251,269,164

Table 2. Statistics

Total Countries:	***Population***
Global Population:	8.16 billion
Average Population:	34.87 million
Median Population:	5.62 million
Population Standard Deviation:	138.35 million

4 DISCUSSION

4.1 Reflections on the Beautiful Soup exercise

During a postmortem about the course between two of the co-authors, one of them decided to try the assignment on their own. They were able to successfully produce a model solution to the assignment independently.

Taking this into consideration puts the student results in a different light. Even though mixing students with different ability levels did not seem to work particularly well in this situation, it is nevertheless considered best practice. A potentially deeper problem was that the students hadn't previously developed good collaborative problem-solving skills.

In our discussions — NB. applying peeragogy at the "curriculum" level — we've produced some ideas about how the project could be run differently next time, hopefully with improved results. They are detailed in Fig. 2. One of the students from the python course was asked why he approached the problem in the way that he did, he responded as follows, "I chose to use Selenium alongside it (Beautiful Soup) due to its ability to handle dynamic content and interact with JavaScript-heavy websites, which is essential for scraping sites that require user interaction. In developing my code, I structured it to first navigate to the target webpage using Selenium, ensuring all elements were fully loaded. Once the data was accessible, I used Beautiful Soup to parse the HTML and extract the required information cleanly. This combination allowed me to efficiently handle both static and dynamic elements of the pages. I scraped datasets from these websites, primarily focusing on data relevant to my project. Through this process I learned about the importance of handling dynamic content and became more proficient in using both Selenium and Beautiful Soup for efficient web scraping." [25]

1. Provide students with an introductory text on relevant aspects of peeragogy, including collaborative problem-solving techniques.
2. Start the project earlier in the semester (even though this was supposed to be the final exam).
3. Ask students to individually choose specific software to create an initial demo of results for the group, and then work to a group consensus about how to complete the project together.
4. Work together to develop further strategies that encourage students to continue to learn independently after the course is complete.

Figure 2. Strategies for improving the Beautiful Soup exercise in future iterations.

4.2 Peeragogy Applications in Teaching English as a First or Second Language

Another exploration of intermixed peeragogy and teaching was done in a few California high school English classrooms. The students were given the opportunity to play a game named "5PH1NX" [14]. There were a series of literary "anomalies" to engage with collaboratively including poems and non-fiction encouraging the students to embrace their own learning adventures. The first cohort thoroughly embraced analyzing the anomalies by writing their own blog posts and even evolving the very rules of the game. This gamification exercise included students taking responsibility for peeragogically assessing each other. They called their assessment approach Project Infinity which they used to independently assign value to the thoughts and activities they deemed worthy. They appreciated being given some power in school rather than solely being told what to do. This cohort even volunteered to

assist with the following year's group of students. This was a positive example of item 4 in Fig. 2 above as the students continued engaging with learning the topic after the course finished. However, the outcome of Project Infinity 2 was less successful, which the teacher hypothesized was because "Project Infinity 2 wasn't theirs. They didn't get to build it. It was handed to them in the same way that a syllabus is handed to them." This reinforces our recommendation in Item 3 of Fig. 2 that students need to co-create the learning experience for peeragogy and pedagogy to successfully co-exist.

A limited example of that co-existence was in an English as a Second Language class for adults in Chicago [16]. The students were instructed to find a way to learn English collaboratively with one another at times when the teacher was not present. To start, they were given a sample of the “Peeragogy in Action” chapter from The Open Book as an introductory text [15] to familiarize themselves with the techniques. Following a class discussion, they took the initiative over their own learning. After considering multiple venues, the students chose to study together on a group Facebook Page. They used it to share resources on grammar concepts from class (such as countable versus uncountable nouns and adverbs of frequency). This group page was also used to share updates on group project work; one student posted about the food they would make for their “Descriptive Cuisine Project”. They ended with 45 posts, which was a clear example of them using English outside of class in public. The understanding was they normally posted on social media in their own language. However, the students did not continue using the Facebook Page once they left the class, which was part of the original goal. The page is no longer available on Facebook, but as of 2014 it had 27 likes [16], and we believe the likes encouraged the students to use the page more while it was operational.

4.3 Other Examples of Peeragogy in Teaching and Learning Contexts

Another attempt to foster peeragogical learning — an early failed experiment — was “DIY Math” [17]. This course was set up to run on the Peer-2-Peer University platform in 2010. Students were asked to study whatever mathematical topic they wanted to look at on their own, and then come together to talk about how it was going in round-table discussions. However, that simply did not work, and attendance quickly dropped to zero. Students need structure and concrete motivating tasks to engage with.

A considerably more successful group learning activity was employed in a course called “Data Science for Design” which ran at the University of Edinburgh. Students were not assumed to know programming techniques and were given introductory programming tutorials as part of the course (though, in fact, some of the students did have a strong background in programming). Students were then teamed up and matched with practical projects in a “data fair,” and worked in groups with mixed abilities to produce a final presentation for the “data holder [18].”

This approach has since been adapted and used to run an extra-curricular Data Challenge series at Oxford Brookes, using data sets from various community stakeholders. So far, much as in the Beautiful Soup exercise, individual students who had a strong programming background were more successful at delivering results in the data challenge. Furthermore, students encountered frustrating non-technical challenges when it came to establishing effective teams. Given that the learning objectives for these exercises include better understanding of “the social side of data science” [19] — part of which is learning how to work well together — these disparate outcomes are concerning. Again, this is similar to the Beautiful Soup project. Nevertheless, the comparative success of “Data Science for Design” illustrates that when the correct incentives and structures are in place — such as a final assessment that tangibly benefits from the input of people with varied skills — group work can be highly effective.

It is worth briefly adding that peeragogy has informed the design of other education-relevant learning activities, such as a workshop with students of education at the technical-professional level, which was intended to help reduce the gender digital divide [20].

5 CONCLUSIONS

Reflection exercises shared between teachers and others with an interest in learning can be quite useful, as we believe this paper confirms. Reflection can also be scaffolded on a large scale. One of the co-authors of this paper has recently been involved in setting up an "open research training community of practice,” for people working in UK universities to reflect on their methods for disseminating open and reproducible research practice within their institutions [21]. In that setting, we are using the Community Canvas from Pfortmüller et al. [22] to structure our co-design activities. While the full apparatus of the Community Canvas would be too heavy for use when co-designing small group learning experiences, student groups could be invited to discuss how they want to learn together. Our Peeragogy patterns,

mentioned earlier, are modular design elements that groups might adopt to further elaborate their ways of working. Additionally, we have recently been considering design strategies that help people engage with the challenges around using AI constructively in the classroom [23].

On a larger scale, initiatives such as UNESCO's 1st International Forum on the Futures of Education [24] highlight the idea that education is crucial for addressing social challenges and transforming the future. In connection with the student results mentioned above, we note that taken together, our countries of residence — the US, UK, and Mexico — comprise only 6.68% percent of the world's population. For many people worldwide, new strategies and infrastructures for effective peer learning could help bridge a considerable opportunity gap.

ACKNOWLEDGEMENTS

Thanks to Charlotte Pierce for her helpful comments that greatly improved this paper. Thanks to Jonathan Lee, Pace University for suggesting Beautiful Soup, which helped inspire this paper.

REFERENCES

[1] M. Tedeschi, (2023). From Classroom to Online Education – An Educator's Insights. Proceedings of the 27th European Conference on Pattern Languages of Programs. Presented at the Irsee, Germany. DOI: https://doi.org/10.1145/3551902.3551977

[2] P. Tyre, (2017) Can a Tech Start-Up Successfully Educate Children in the Developing World? The New York Times Magazine. June 27, 2017.

[3] J. Corneli, C.J. Danoff, C. Pierce, P. Ricaurte, and L. Snow MacDonald, 2015. Patterns of Peeragogy. In: Pattern Languages of Programs Conference 2015 (PLoP'15), Pittsburgh, PA, USA, October 24-26, 2015. Ed. by F. Correia. ACM. 2016.

[4] Wikipedia contributors. (2024, May 2). Zone of proximal development. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Zone_of_proximal_development&oldid=1221950657

[5] J. Corneli, A. Murphy, R.S. Puzio, L. Vivier, N. Alhasan, C.J. Danoff, V. Bruno, and C. Pierce, (2022). Patterns of patterns: A methodological reflection on the future of design pattern methods. In S. Inayatullah, R. Mercer, I. Milojević, & J. A. Sweeney (Eds.), CLA 3.0: Thirty Years of Transformative Research. Tamkang University Press.

[6] J. Corneli, N. Alhasan, L. Vivier, A. Murphy, R.S. Puzio, A. Tabor, S. Ayloo, C. Pierce, C.J. Danoff, M. Tedeschi, M. Singh, and K. Khetan (2023). Patterns of Patterns II. To appear in Proceedings of Pattern Languages of Programs 2023.

[7] "Beautiful Soup Documentation", Accessed 20 September, 2024. Retrieved from https://beautiful-soup-4.readthedocs.io/en/latest/

[8] "Python 3.12.6 documentation", Accessed 20 September, 2024. Retrieved from https://docs.python.org/3/

[9] "pandas documentation", Accessed 20 September, 2024. Retrieved from https://pandas.pydata.org/docs/index.html

[10] "Master-Student Collaborations in Data Visualization", Accessed 20 September, 2024. Retrieved from https://datafairs.github.io/

[11] "Matplotlib 3.9.2 documentation", Accessed 20 September, 2024. Retrieved from https://matplotlib.org/stable/index.html

[12] "Selenium with Python", Accessed 20 September, 2024. Retrieved from https://selenium-python.readthedocs.io

[13] "Project Jupyter Documentation", Accessed 20 September, 2024. Retrieved from https://docs.jupyter.org/en/latest/

[14] "Visual Studio documentation , Accessed 20 September, 2024. Retrieved from https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022

[15] “5PH1NX: 5tudent Peer Heuristic for 1Nformation Xchange“, Accessed 20 September, 2024. Retrieved from https://peeragogy.org/5ph1nx

[16] J. Corneli, C.J. Danoff, A. Keune, and A. Lyons (2013). Peeragogy in Action. In: The Open Book. Ed by Jussi Nissila; Kaitlyn Braybrooke; Timo Vuorikivi. Published by The Finnish Institute in London ISBN: 978-0-9570776-3-8. https://openbook.okfn.org

[17] C.J. Danoff. (2014). Implementing Communicative Language Teaching in an ESL Classroom with Peeragogy on Facebook, https://neltaeltforum.wordpress.com/2014/09/02/335/

[18] J. Corneli, and C.J. Danoff, (2011) Paragogy. In Proceedings of the 6th Open Knowledge Conference. Ed. by Auer, S, https://ceur-ws.org/Vol-739/paper_5.pdf

[19] “Master-Student Collaborations in Data Visualization “, Accessed 20 September, 2024. Retrieved from https://datafairs.github.io/

[20] J. Corneli, D. Murray-Rust, and B. Bach (2018). Towards Open-World Scenarios: Teaching the Social Side of Data Science. In Cybernetic Serendipity Reimagined Symposium, Proc. Annual Convention of the Society for the Study of Artificial Intelligence and Simulation of Behaviour http://aisb2018.csc.liv.ac.uk/PROCEEDINGS%20AISB2018/Cybernetic%20Serendipity%20Reimagined%20-%20AISB2018.pdf#page=25

[21] C.E. George-Reyes, I.C. Peláez-Sánchez, and L.D. Glasserman-Morales (2024). Digital Environments of Education 4.0 and complex thinking: Communicative Literacy to close the digital gender gap. Journal of Interactive Media in Education, 2024(1):3, pp. 1–20. DOI: https://doi.org/10.5334/jime.833

[22] J. Corneli, S. Boneham, and N. Jacobs (2024) A community of practice for open research trainers. Poster presented at Second International Research Culture Conference. http://metameso.org/~joe/docs/warwick-ircc2024.pdf

[23] F. Pfortmüller, N. Luchsinger, and S. Mombartz (2017). Community canvas, https://community-canvas.org/

[24] M. Tedeschi, P. Ricaurte, S. Ayloo, J.Corneli, C.J. Danoff, S. Belich, (forthcoming). AI Future Envisioning with PLACARD. Proceedings of the 28th European Conference on Pattern Languages of Programs. Presented at the Irsee, Germany.

[25] “UNESCO International Forum on the Futures of Education”, https://www.unesco.org/en/renewing-education-transform-future

[26] Pulagam, Durga Avinash Reddy,(dp49842n@pace.edu).Today’s class IS-623.[Personal email, 1 Nov 2024] to M. Tedeschi(mtedeschi@pace.edu).

APPENDIX

Figures

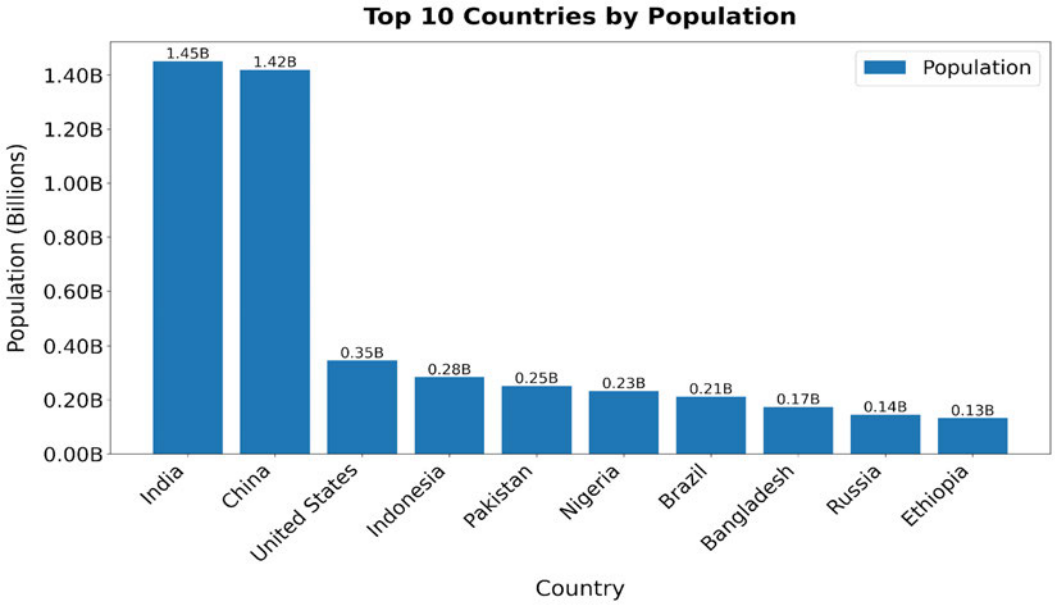

Figure 4. Top 10 countries by population

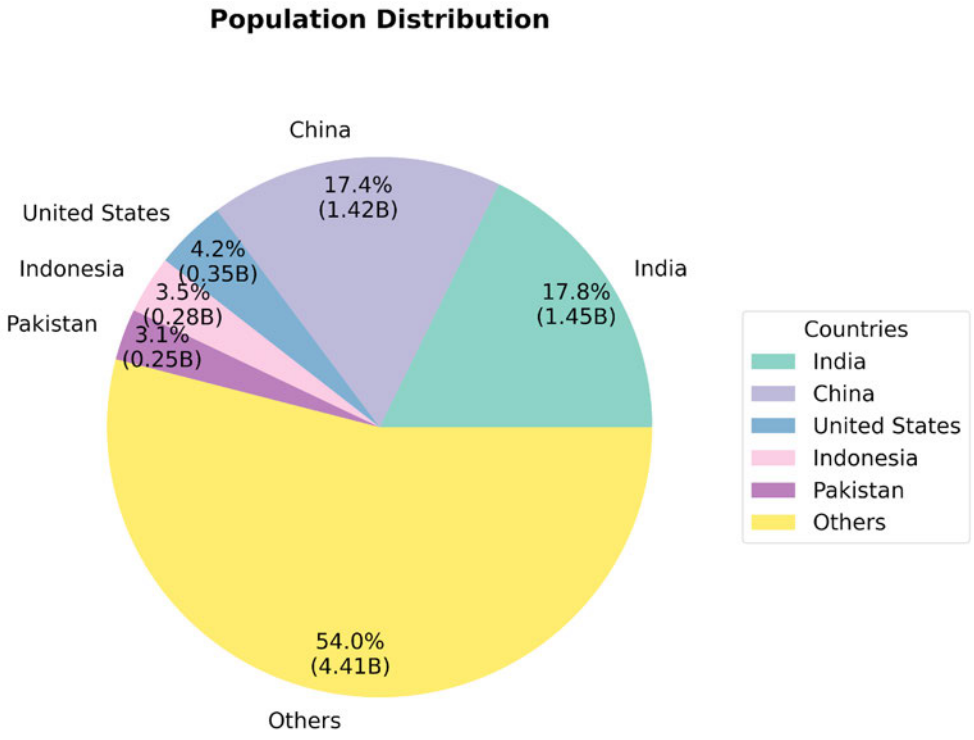

Figure 5. Population Distribution (Top 5 Countries VS Others)

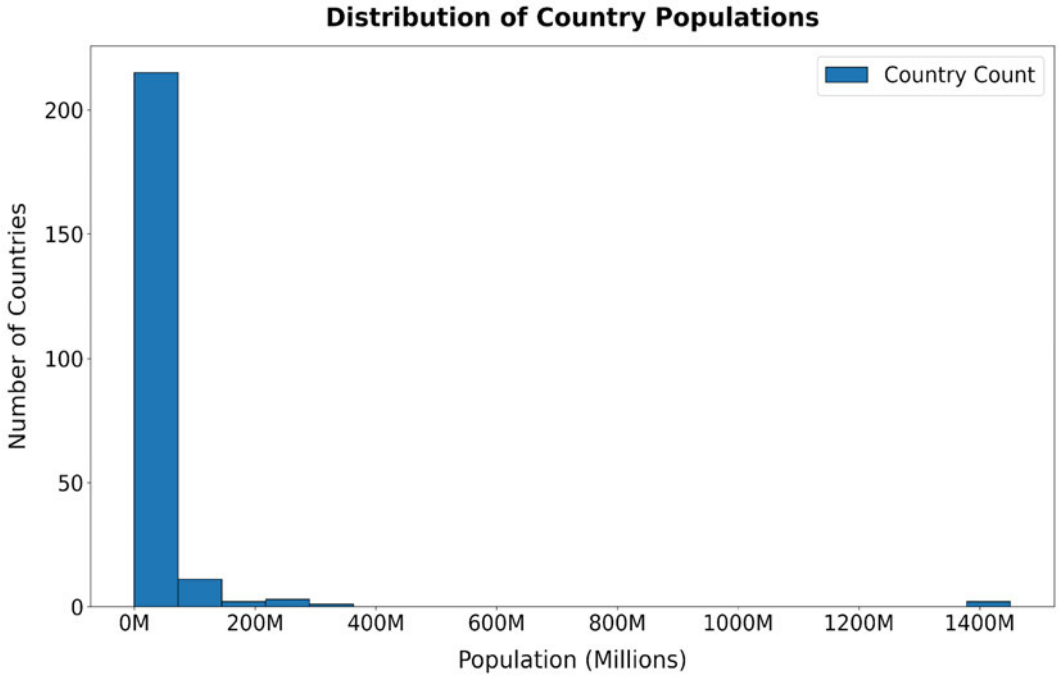

Figure 6. Distribution of Country Populations

5.1 Script

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

def scrape_population_data():
    url = "https://www.worldometers.info/world-population/population-by-country/"
    response = requests.get(url)
    if response.status_code != 200:
        print(f"Failed to retrieve the webpage. Status code: {response.status_code}")
        return None
    soup = BeautifulSoup(response.content, 'html.parser')
    table = soup.find('table', {'id': 'example2'})
    if not table:
        print("Could not find the population table on the webpage.")
        return None
    data = []
    for row in table.find_all('tr')[1:]:  # Skip header row
        cols = row.find_all('td')
        if len(cols) >= 3:
            country = cols[1].text.strip()
            population = cols[2].text.strip().replace(',', '')
            data.append([country, int(population)])
    df = pd.DataFrame(data, columns=['Country', 'Population'])
    return df

def calculate_statistics(df):
    stats = {
        'Total Countries': len(df),
        'Global Population': f"{df['Population'].sum() / 1e9:.2f} billion",
        'Average Population': f"{df['Population'].mean() / 1e6:.2f} million",
        'Median Population': f"{df['Population'].median() / 1e6:.2f} million",
        'Population Standard Deviation': f"{df['Population'].std() / 1e6:.2f} million",
```

```
        'Most Populous Country': f"{df.loc[df['Population'].idxmax(), 'Country']} ({df['Population'].max() /
1e9:.2f} billion)",
        'Least Populous Country': f"{df.loc[df['Population'].idxmin(), 'Country']} ({df['Population'].min() /
1e3:.2f} thousand)",
    }
    return stats

def set_larger_fonts():
    plt.rcParams['font.size'] = 22
    plt.rcParams['axes.titlesize'] = 26
    plt.rcParams['axes.labelsize'] = 24
    plt.rcParams['xtick.labelsize'] = 22
    plt.rcParams['ytick.labelsize'] = 22
    plt.rcParams['legend.fontsize'] = 22
    plt.rcParams['figure.titlesize'] = 30

def create_bar_chart(df):
    set_larger_fonts()
    plt.figure(figsize=(16, 10))
    top_10 = df.nlargest(10, 'Population')
    bars = plt.bar(top_10['Country'], top_10['Population'] / 1e9)
    plt.title('Top 10 Countries by Population', fontweight='bold', pad=20)
    plt.xlabel('Country', labelpad=15)
    plt.ylabel('Population (Billions)', labelpad=15)
    plt.xticks(rotation=45, ha='right')
    plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda x, p: f'{x:.2f}B'))
    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2., height,
                 f'{height:.2f}B',
                 ha='center', va='bottom', fontsize=16)
    plt.legend(['Population'], loc='upper right')
    plt.tight_layout()
    plt.savefig('top_10_population_bar_chart.png', dpi=300, bbox_inches='tight')
    plt.close()

def create_pie_chart(df):
    set_larger_fonts()
    plt.figure(figsize=(14, 12))
    top_5 = df.nlargest(5, 'Population')
    others = pd.DataFrame({'Country': ['Others'],
                           'Population': [df['Population'].sum() - top_5['Population'].sum()]})
    pie_data = pd.concat([top_5, others])
    colors = plt.cm.Set3(np.linspace(0, 1, len(pie_data)))
    wedges, texts, autotexts = plt.pie(pie_data['Population'], labels=pie_data['Country'],
            autopct=lambda pct: f'{pct:.1f}%\n({pct/100.*df["Population"].sum()/1e9:.2f}B)',
            textprops={'fontsize': 22}, pctdistance=0.85, colors=colors)
    plt.title('Population Distribution', fontweight='bold', pad=20)
    plt.legend(wedges, pie_data['Country'],
               title="Countries",
               loc="center left",
               bbox_to_anchor=(1, 0, 0.5, 1))
    plt.axis('equal')
    plt.tight_layout()
    plt.savefig('population_distribution_pie_chart.png', dpi=300, bbox_inches='tight')
    plt.close()

def create_histogram(df):
    set_larger_fonts()
    plt.figure(figsize=(16, 10))
```

```
    df['Population (Millions)'] = df['Population'] / 1e6
    plt.hist(df['Population (Millions)'], bins=20, edgecolor='black')
    plt.title('Distribution of Country Populations', fontweight='bold', pad=20)
    plt.xlabel('Population (Millions)', labelpad=15)
    plt.ylabel('Number of Countries', labelpad=15)
    plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(lambda x, p: f'{x:.0f}M'))
    plt.legend(['Country Count'], loc='upper right')
    plt.tight_layout()
    plt.savefig('population_distribution_histogram.png', dpi=300, bbox_inches='tight')
    plt.close()

def main():
    print("Scraping population data from Worldometer...")
    df = scrape_population_data()
    if df is not None and not df.empty:
        print("\nSample of scraped data:")
        print(df.head().to_string())

        stats = calculate_statistics(df)
        print("\nStatistics:")
        for key, value in stats.items():
            print(f"{key}: {value}")
        create_bar_chart(df)
        create_pie_chart(df)
        create_histogram(df)
        print("\nCharts have been saved as:")
        print("1. top_10_population_bar_chart.png")
        print("2. population_distribution_pie_chart.png")
        print("3. population_distribution_histogram.png")
        df.to_csv('world_population_data.csv', index=False)
        print("\nData has been saved to 'world_population_data.csv'")
    else:
        print("Failed to process data.")

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