Pyneapple-R: Scalable and Expressive Spatial Regionalization

Yunfan Kang*† Yongyi Liu‡§ Hussah Alrashid‡§ Akash Bilgi‡ Siddhant Purohit‡
Ahmed Mahmood¶ Sergio Rey‖** Amr Magdy‡§
* *CyberGIS Center for Advanced Digital and Spatial Studies, University of Illinois at Urbana-Champaign*
† *Department of Geography and Geographic Information Science, University of Illinois at Urbana-Champaign*
‡ *Department of Computer Science and Engineering, University of California, Riverside*
§ *Center for Geospatial Sciences, University of California, Riverside*
¶*Google LLC* ‖ *Department of Geography, San Diego State University*
** *Center for Open Geographical Science, San Diego State University*
Email: yfkang@illinois.edu {yliu786, halra004, abilg003}@ucr.edu
siddhant.0519@gmail.com amahmoo@google.com srey@sdsu.edu amr@cs.ucr.edu

***Abstract*—This paper demonstrates *Pyneapple-R*, an open-source library for scalable and expressive regionalization. Regionalization algorithms, also known as the 'spatially-constrained clustering algorithms', have been widely adopted in spatial analysis tasks and now evolving towards a more large-scale and fine-scale direction. Through collaborations with social scientists and domain experts, we have identified emerging challenges in existing regionalization techniques, particularly regarding scalability and expressiveness. As data volumes continue to grow and regionalization algorithms become increasingly crucial to decision-making across various fields, enhancing these aspects can significantly impact the quality and effectiveness of research and applications. To address these challenges, *Pyneapple-R* provides novel algorithms for regionalization queries including the expressive p-regions algorithm, the scalable max-p regions algorithm, and the expressive max-p regions problem. To showcase *Pyneapple-R*, we have developed frontend web applications that enable users to interact with the algorithms by selecting constraints or simply engaging in conversation with the system to issue queries with the help of popular AI models. Interactive notebooks, designed to demonstrate the superiority and simplicity of *Pyneapple-R*, provide varying levels of detail to help social scientists and developers explore its full potential.**

***Index Terms*—spatial analysis, algorithms, library, regionalization, query**

I. Introduction

Spatial analysis has become an indispensable tool in various social science studies [1]–[3]. It enables social scientists to conduct complex analyses on diverse socioeconomic phenomena [4] and plays a vital role in everyday applications and policy making. Among the spatial analysis techniques, regionalization has gained much attention in recent years [5]. Spatial regionalization involves grouping spatial areas into spatially contiguous and homogeneous regions. It finds applications in numerous fields, such as epidemic analysis [6], service delivery systems [7], water quality assessment [8], weather temperature classification [9], rainfall erosivity estimation [10], health data analysis [11], spatial crowdsourcing [12], [13], and constituency allocation [14]. Regionalization is also one of the core functionalities of the spatial analysis module of the geographic information systems (GIS) and is being used by multitudes of users and organizations every day [15].

Although regionalization problems can be modeled as mixed-integer problems and solved using general MIP solvers [16], deriving an optimal solution in a feasible timeframe is formidable, especially for larger inputs. This complexity stems from their NP-hard nature. As a result, several heuristic algorithms have been proposed to address the regionalization problem [5], [17]. However, despite the benefits social scientists gain from regionalization techniques, particularly with the increased availability of big geospatial data, several emerging problems remain unsolved due to the limited scalability and expressiveness of existing tools and techniques. For instance, the current best-performing algorithm, max-p-regions (MP-regions) — available in the PySAL library and ArcGIS — can take 3-10s of minutes to process a dataset with approximately 3K areas, depending on the threshold value, limiting its practical applicability. Moreover, the traditional MP-regions formulation often falls short in catering to multifaceted analytical tasks. While it permits a *singular*, user-defined *threshold lower bound on one attribute*, many real-world scenarios necessitate multiple constraints to draw meaningful insights. An example can be seen in the context of COVID-19 where transmission patterns are intertwined with factors like prosperity and mobility [6]. The existing formulation's inability to address such composite queries impedes the broader adoption of regionalization algorithms.

This paper presents a system demonstration of *Pyneapple-R*[1], an open-source library for scalable and expressive regionalization, currently under active development with more features being added. *Pyneapple-R* builds upon our research work, which has yielded three novel algorithms: the scalable p-regions with user-defined constraints (PRUC) [18], the scalable MP-regions algorithm (SMP) [19], [20], and the expressive MP-regions algorithm (EMP) [21]. Attendees at our demonstration can acquaint themselves with *Pyneapple-R* through

[1]https://github.com/MagdyLab/Pyneapple

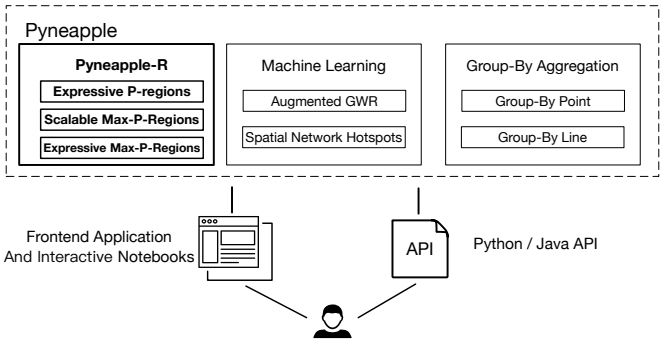

Fig. 1. *Pyneapple* Overview

frontend applications, which allow interactive constraint selection and visualization of results. In a novel approach, the GPT-3.5 model enables users to converse directly with the system, while the backend automates query interpretation and processing. Additionally, interactive Jupyter Notebooks for Python and Java enthusiasts will showcase API usage and the edge our algorithms have over existing counterparts. Subsequent sections offer an in-depth exploration of the *Pyneapple-R* library (Section II) and the demonstration specifics (Section III).

II. *Pyneapple-R* Overview

Pyneapple-R is an integral sub-package of the more extensive *Pyneapple* library [22]. Figure 1 furnishes a comprehensive overview of the *Pyneapple* ecosystem. The current version of *Pyneapple* comprises three main modules: regionalization queries, machine learning (ML) assisted analysis and group-by-aggregation queries. *Pyneapple-R* refers to the regionalization module, as bolded in the figure. Each of the algorithms in *Pyneapple-R* is equipped with thorough Python and Java API documentation, facilitating a seamless integration into the broader data science landscape. The rest of this section delves deeper into the specifics of each algorithm housed within the *Pyneapple-R* library.

A. Expressive P-Regions Problem

The p-regions problem is akin to clustering tasks like k-means, where users must specify the number of regions and an optimization objective. However, existing variants of the p-regions problem lack support for user-defined constraints, limiting its applicability across various domains and a wide range of use cases. To address this limitation, the **PRUC** problem [18] generalizes the p-regions problem to include an additional user-defined constraint, such as a minimum population requirement for each region.

Introducing a user-defined constraint in PRUC presents several challenges. First, existing techniques have a significant probability, up to 80%, of generating regions that fall short of the user-specified constraints. Second, ensuring that regions adhere to the constraints often demands extensive spatial rearrangements. Third, the additional overhead in producing valid solutions exacerbates scalability issues, making it more challenging to handle large datasets.

To adeptly navigate these challenges, the *Pyneapple-R* library introduces the Global Search with Local Optimization (GSLO) algorithm. GSLO operates in two phases (1) Global Search and (2) Local Optimization. The Global Search phase employs innovative techniques to derive a region partitioning that aligns with the user-defined constraints, boasting a high success probability. In the Local Optimization phase, parallel stages are used to progressively improve the partitioning's quality concerning similarity properties within each region. Experimental results demonstrate that GSLO is over 100× faster and achieves up to 6× better heterogeneity compared to state-of-the-art algorithms. Furthermore, GSLO solves the original p-regions problem with up to 4× better heterogeneity than existing algorithms.

B. Scalable MaxP-Regions Problem

In contrast to the conventional p-regions problem, the MP-regions problem deviates by requiring users to specify a constraint on an attribute with a lower bound threshold, such as a minimum total population. This constraint-based approach allows for the meaningful definition of spatial regions based on attribute values, rather than requiring the explicit specification of the number of regions. It proves especially valuable when social scientists aim to automatically determine the appropriate spatial scale, as manually specifying the number of regions can be challenging. Unfortunately, scalability issues have historically hindered the effective use of MP-regions, particularly when dealing with a large number of geographical units [23]–[25].

Scaling up MP-regions confronts several formidable challenges. First, MP-regions is an NP-hard problem, rendering exhaustive exploration of all possible solutions and finding an optimal solution prohibitively costly. Second, applying standard spatial partitioning methods to divide input spatial areas into smaller subsets while preserving spatial contiguity is far from straightforward. Conventional techniques tend to partition spatial objects based solely on their individual boundaries, often ignoring the spatial relationships with neighboring objects. This approach leads to spatially disconnected partitions, thwarting parallelized methods from generating connected regions. Third, existing techniques for MP-regions typically employ tightly coupled steps that challenges straightforward parallelization. Consequently, these methods remain inherently centralized and struggle to support large datasets effectively.

To navigate these impediments, *Pyneapple-R* unveils the **SMP** strategy [19], [20], which innovatively refines the region-building process through parallelization. SMP's cornerstone is its two-phase spatial partitioning approach that emphasizes spatial contiguity and minimizes processing overhead. Such a blueprint paves the way for efficient, fully parallelized methodologies, adeptly scaling with augmented computational resources on commodity machines. As a result, SMP achieves an impressive 97% reduction in query time and extends its support to datasets that are an order of magnitude larger than what state-of-the-art approaches can accommodate.

C. Expressive MaxP-Regions Problem

In addition to the scalability issue pointed out in Section II-B, the MP-regions formulation is also limited in expressiveness because it only supports a single constraint with a lower-bound threshold. However, many real-world analysis tasks require multiple constraints with different aggregate functions. To address the need for a broader range of complex analysis tasks, the **EMP** formulation [21] was introduced. EMP extends the MP-regions problem by incorporating the five SQL aggregation operators: MIN, MAX, AVG, SUM, and COUNT. Notably, it accommodates multiple constraints within a singular query. Each constraint is adaptable via a range operator, enabling representation of both lower and upper bounds.

This substantial enhancement of the expressiveness of the MP-regions problem presents several challenges. First, EMP is computationally more demanding than MP-regions. It formulates the MP-regions problem as a subproblem by specifying only a single SUM constraint with a lower-bound threshold. Additionally, supporting multiple constraints with varying mathematical properties and range operators renders existing MP-regions solutions inadequate for EMP. Second, the new constraints are non-monotonic, meaning that adding or removing areas to or from a region does not guarantee satisfaction or violation of these constraints. This necessitates exhaustive exploration of the entire search space, which is infeasible for such an NP-hard problem. Third, while satisfying multiple non-monotonic constraints, a region can easily become oversized, reducing the number of regions (p), which contradicts the goal of maximizing p in the original definition of MP-regions.

The *Pyneapple-R* offers a three-phase solution, namely *FaCT* algorithm, to address these challenges and solve the EMP problem. The first phase establishes theoretical bounds to assess the feasibility of finding a solution given the user-defined constraints. The second phase constructs a feasible solution that adheres to the input constraints while maximizing the number of regions (p). This phase comprises three independent steps, each tailored to leverage the mathematical properties of a specific constraint type to maximize p to its theoretical upper limit. The third phase employs a local search based on the Tabu search [23] algorithm to enhance overall region heterogeneity. Notably, our algorithm demonstrates exceptional scalability when processing datasets larger than those previously examined in the existing literature as well.

III. Demonstration Scenario

To demonstrate *Pyneapple-R*, we design different scenarios for different groups of target audiences. Users will interact with the *Pyneapple-R* library through user-friendly web applications and Jupyter Notebooks, illustrating the ease of use and interactive nature of the toolkit. Attendees will be able to visualize the results, explore different parameter settings, and gain a deep understanding of the potential applications of *Pyneapple-R* in various domains. By the end of this demonstration scenario, attendees will be confident in the capabilities of *Pyneapple-R* to address complex spatial analysis tasks and appreciate the potential impact it can have on real-world decision-making processes.

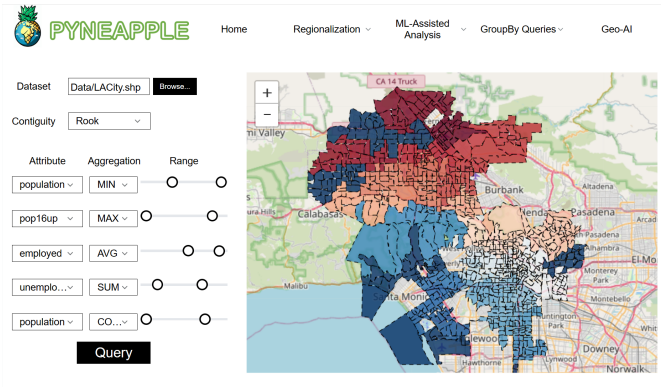

Fig. 2. Frontend UI for Regionalization Queries

A. Scenario 1: Interactive Querying and Visualization

To demonstrate its capabilities and engage a general audience with varying technical backgrounds, we have developed an intuitive and user-friendly frontend that offers a straightforward web UI. Users can easily issue queries by selecting datasets and composing constraints by interacting with dropdown menus and sliders, as illustrated in Figure 2. Let's assume a user is interested in exploring the employment situation in Los Angeles. By clicking the *Browse* button, the shapefile of the census tracts of Los Angeles is loaded and visualized in the left panel. Simultaneously, the contiguity matrices are computed at the back end so that the user can select how he wants to define the connectivity of the regions. After specifying the attribute names and the aggregation methods, the slide bars are initialized, allowing users to set the range for each constraint intuitively. Upon pressing the *Query* button, the query is sent to the backend, and the corresponding regionalization algorithm (in this case, the EMP module) performs the regionalization and returns the region labels. The regions are then automatically colored, and the users can interact with the visualization to view the detailed attribute values by clicking on the regions. For use cases whose scalability is greatly improved by *Pyneapple-R*, such as the SMP, we also include the APIs of the state-of-the-art implementations. The frontend application also provides an option to turn on the side-by-side comparison with statistics to illustrate the superiority and the correctness of *Pyneapple-R* algorithms. This demonstration scenario highlights the potential of *Pyneapple-R* in making spatial analysis more accessible and efficient for various users, emphasizing its ability to handle an enriched set constraints and large-scale problems.

B. Scenario 2: Building Applications

For social scientists working primarily in the Python environment, *Pyneapple-R* encapsulates all low-level Java implementations as black boxes and exposes just the essential functions as plug-and-use modules.

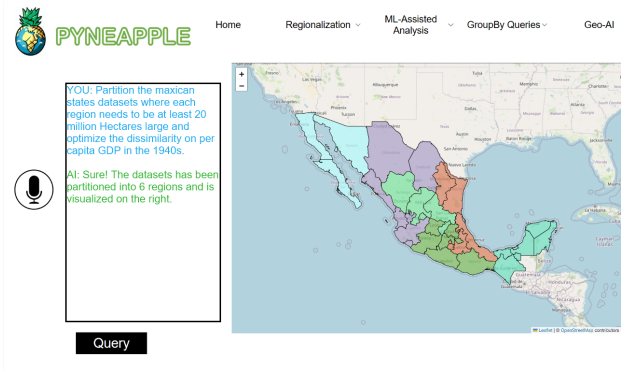

Fig. 3. Query by Talking to the System

To demonstrate the simplicity of *Pyneapple-R* Python APIs, we carefully design Jupyter Notebooks for each of the modules. Each notebook contains detailed guidance and examples to walk the audience through the life cycles of algorithms in *Pyneapple-R*. To get started, we show that social scientists can simply install the *Pyneapple-R* Python package through conda. All dependencies, including the JDK and the configuration of the Java environment, can be configured by the script and the users will not be aware of the Java-Python bridge. For demonstrating the usage of *Pyneapple-R* Python APIs, we adopt familiar, classic, or popular use cases that social scientists can relate to in order to demonstrate how the APIs of the existing techniques can be replaced with *Pyneapple-R* APIs that accept the same input and produce compatible output within the original computation workflow. Comparisons between *Pyneapple-R* APIs and original APIs, along with corresponding visualizations, are shown to highlight *Pyneapple-R*'s superiority in terms of efficiency and result quality. Through this demonstration, we show users that their work is painlessly improved with *Pyneapple-R* APIs while gaining huge benefits in scalability and expressiveness.

C. Scenario 3: Natural Language Querying

To further ease using *Pyneapple-R* for social scientists and extend its interactivity and user friendliness, we introduce a middle layer powered by GPT-3.5 APIs to enable the audiences to interact with the system using natural language. This innovative approach allows users to either speak their query or type it in as text, making the process more user-friendly and intuitive. For example, a user may say, "Partition the Mexican states datasets where each region needs to be at least 20 million Hectares large and optimize the dissimilarity on per capita GDP in the 1940s.", as shown in Figure 3 The recorded audio is then processed by the large-v2-whisper model to generate text. With the help of the text-davinci-003 model, the AI-driven system would recognize the query, identify the required dataset and parameters to invoke the corresponding function, and execute the necessary steps to visualize the results on the frontend. The user can modify the query to get a more desirable result or issue another query by simply continuing the discussion.

REFERENCES

[1] J. R. Logan, "Making a place for space: Spatial thinking in social science," *Annual review of sociology*, vol. 38, pp. 507–524, 2012.

[2] D. Darmofal, *Spatial analysis for the social sciences*. Cambridge University Press, 2015.

[3] L. Anselin, "The future of spatial analysis in the social sciences," *Geographic information sciences*, vol. 5, no. 2, pp. 67–76, 1999.

[4] L. M. Scott and M. V. Janikas, "Spatial statistics in arcgis," in *Handbook of applied spatial analysis: Software tools, methods and applications*, pp. 27–41, Springer, 2009.

[5] R. Wei, S. Rey, and E. Knaap, "Efficient Regionalization for Spatially Explicit Neighborhood Delineation," *IJGIS*, vol. 35, pp. 1–17, 2020.

[6] R. Benedetti, F. Piersimoni, G. Pignataro, and F. Vidoli, "The Identification of Spatially Constrained Homogeneous Clusters of Covid-19 Transmission in Italy," *RSPP*, vol. 12, pp. 1169–1187, 2020.

[7] M. P. Armstrong, G. Rushton, R. Honey, B. T. Dalziel, P. Lolonis, S. De, and P. J. Densham, "Decision Support for Regionalization: A Spatial Decision Support System for Regionalizing Service Delivery Systems," *CEUS*, vol. 15, pp. 37–53, 1991.

[8] K. S. Cheruvelil, P. A. Soranno, M. T. Bremigan, T. Wagner, and S. L. Martin, "Grouping Lakes for Water Quality Assessment and Monitoring: The Roles of Regionalization and Spatial Scale," *JEM*, vol. 41, pp. 425–440, 2008.

[9] A. El Kenawy, J. I. López-Moreno, and S. M. Vicente-Serrano, "Summer Temperature Extremes in Northeastern Spain: Spatial Regionalization and Links to Atmospheric Circulation (1960–2006)," *TAC*, vol. 113, pp. 387–405, 2013.

[10] S. Schönbrodt-Stitt, A. Bosch, T. Behrens, H. Hartmann, X. Shi, and T. Scholten, "Approximation and Spatial Regionalization of Rainfall Erosivity Based on Sparse Data in a Mountainous Catchment of the Yangtze River in Central China," *JESPR*, vol. 20, pp. 6917–6933, 2013.

[11] N. Bullen, G. Moon, and K. Jones, "Defining localities for health planning: a gis approach," *Social Science & Medicine*, vol. 42, no. 6, pp. 801–816, 1996.

[12] Z. Chen, P. Cheng, L. Chen, X. Lin, and C. Shahabi, "Fair task assignment in spatial crowdsourcing," *PVLDB*, vol. 13, no. 12, pp. 2479–2492, 2020.

[13] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, "Trichromatic online matching in real-time spatial crowdsourcing," in *ICDE*, pp. 1009–1020, IEEE, 2017.

[14] D. J. Rossiter and R. J. Johnston, "Program group: the identification of all possible solutions to a constituency-delimitation problem," *Environment and Planning A*, vol. 13, no. 2, pp. 231–238, 1981.

[15] Esri, "Unlock your data with machine learning and clustering tools in arcgis pro." https://mediaspace.esri.com/media/t/1_ghu5dirn, 2018. Accessed on Sep 26, 2023.

[16] J. C. Duque, R. L. Church, and R. S. Middleton, "The p-regions problem.," *Geographical Analysis*, vol. 43, no. 1, pp. 104–126, 2011.

[17] J. C. Duque, L. Anselin, and S. J. Rey, "The max-p-regions problem," *Journal of Regional Science*, vol. 52, no. 3, pp. 397–419, 2012.

[18] Y. Liu, A. R. Mahmood, A. Magdy, and S. Rey, "PRUC: P-regions with User-defined Constraint," *PVLDB*, vol. 15, no. 3, pp. 491–503, 2021.

[19] H. Alrashid, Y. Liu, and A. Magdy, "SMP: Scalable Max-P Regionalization," in *ACM SIGSPATIAL*, pp. 1–4, 2022.

[20] H. Alrashid, Y. Liu, and A. Magdy, "Page: Parallel scalable regionalization framework," *TSAS*, vol. 9, no. 3, pp. 1–26, 2023.

[21] Y. Kang and A. Magdy, "EMP: Max-P Regionalization with Enriched Constraints," in *IEEE ICDE*, pp. 1914–1926, 2022.

[22] MagdyLab, "Pyneapple-r." https://github.com/MagdyLab/Pyneapple.

[23] A. Poorthuis, "How to draw a neighborhood? the potential of big data, regionalization, and community detection for understanding the heterogeneous nature of urban neighborhoods," *Geographical Analysis*, vol. 50, no. 2, pp. 182–203, 2018.

[24] R. Wei, S. Rey, and E. Knaap, "Efficient regionalization for spatially explicit neighborhood delineation," *International Journal of Geographical Information Science*, vol. 35, no. 1, pp. 135–151, 2021.

[25] S. E. Spielman and D. C. Folch, "Reducing uncertainty in the american community survey through data-driven regionalization," *PloS one*, vol. 10, no. 2, p. e0115626, 2015.