LIBKDV: A Versatile Kernel Density Visualization Library for Geospatial Analytics

Tsz Nam Chan
Hong Kong Baptist University
edisonchan@comp.hkbu.edu.hk

Pak Lon Ip
University of Macau
paklonip@um.edu.mo

Kaiyan Zhao
University of Macau
mc14977@um.edu.mo

Leong Hou U
University of Macau
ryanlhu@um.edu.mo

Byron Choi
Hong Kong Baptist University
bchoi@comp.hkbu.edu.hk

Jianliang Xu
Hong Kong Baptist University
xujl@comp.hkbu.edu.hk

ABSTRACT

Kernel density visualization (KDV) has been widely used in many geospatial analysis tasks, including traffic accident hotspot detection, crime hotspot detection, and disease outbreak detection. Although KDV can be supported by many scientific, geographical, and visualization software tools, none of these tools can support high-resolution KDV with large-scale datasets. Therefore, we develop the first versatile programming library, called LIBKDV, based on the set of our complexity-optimized algorithms. Given the high efficiency of these algorithms, LIBKDV not only accelerates the KDV computation but also enriches KDV-based geospatial analytics, including bandwidth-tuning analysis and spatiotemporal analysis, which cannot be natively and feasibly supported by existing software tools. In this demonstration, participants will be invited to use our programming library to explore interesting hotspot patterns on large-scale traffic accident, crime, and COVID-19 datasets.

PVLDB Reference Format:
Tsz Nam Chan, Pak Lon Ip, Kaiyan Zhao, Leong Hou U, Byron Choi, and Jianliang Xu. LIBKDV: A Versatile Kernel Density Visualization Library for Geospatial Analytics. PVLDB, 15(12): 3606-3609, 2022.
doi:10.14778/3554821.3554855

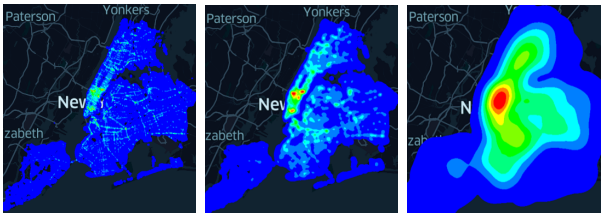
1 INTRODUCTION

Kernel density visualization (KDV) [9, 10, 13] is an important tool for geospatial analytics, which has been extensively used in many location-based applications. Some representative examples include traffic accident hotspot detection [17, 22], crime hotspot detection [15, 19], and disease outbreak detection [14, 23]. Due to its wide range of applications, KDV is supported by default in many popular geographical software tools, e.g., QGIS [7] and ArcGIS [1], scientific computing tools, e.g., Scipy [21], Statsmodels [20], and Scikit-learn [18], and visualization tools, e.g., KDV-Explorer [12] and Deck.gl [3]. Despite this, none of these tools can be scalable to support high-resolution KDV (e.g., 1280×960) for large-scale location datasets (e.g., one million data points) due to the high computational cost. Besides, the computational cost limits the applicability of using the off-the-shelf software tools to support advanced (or more complex) geospatial analytics, e.g., bandwidth-tuning analysis and spatiotemporal analysis, which involves computing multiple KDVs in one batch.

To overcome this issue, we ask a question: *Can we develop the first library that can reduce the worst-case time complexity for supporting different types of KDV-based geospatial analytics?* In order to answer this question, we develop the first complexity-optimized programming library, called LIBKDV, by adopting the new camp of KDV algorithms, including SLAM [13] and SWS [11], in our recent studies. Better still, we fully parallelize our KDV algorithms, SLAM and SWS, to further optimize the efficiency of LIBKDV.

With the complexity-optimized LIBKDV, we enable more advanced geospatial analytics which cannot be natively and feasibly supported by existing software tools, especially for using large-scale datasets and high resolutions. In this paper, we use LIBKDV to demonstrate two representative examples of KDV-based geospatial analytics, which are summarized as follows.

Bandwidth-tuning analysis: The visualization quality of KDV significantly depends on the bandwidth parameter b [10] (which will be discussed in detail in Section 2.1). If we choose a small bandwidth b (cf. Figure 1a), we cannot detect any hotspot region (i.e., undersmoothing). On the other hand, the hotspot regions tend to be very large (i.e., oversmoothing) if we choose a large bandwidth b (cf. Figure 1c). To obtain the hotspot map with the most suitable bandwidth b (cf. Figure 1b), domain experts need to generate multiple KDVs with different bandwidths, i.e., b_1, b_2,..., b_L, and choose the most suitable one [10, 24].



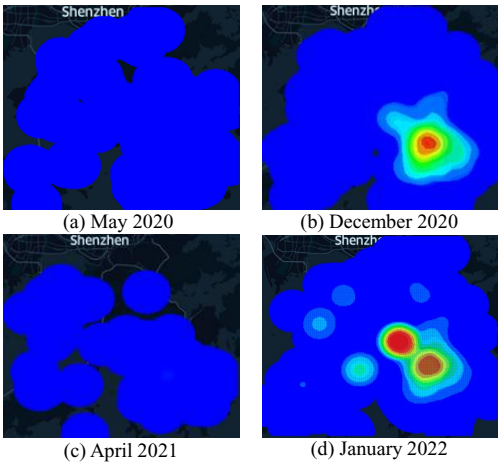
(a) Small b (b) Moderate b (c) Large b

Figure 1: Traffic accident hotspot maps in New York using different bandwidths b.

Spatiotemporal analysis: Many geographical phenomena (e.g., disease outbreak) depend on both location and time. Using Figure 2 as an example, observe that the COVID-19 pandemic in Hong Kong is more serious in December 2020 and January 2022. Therefore, it would be interesting to understand how the hotspots change with respect to different timestamps [14, 15, 17]. To achieve this

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554855

goal, domain experts can leverage a more complex spatiotemporal kernel density function (which will be discussed in Section 2.1) to generate time-dependent hotspot maps which correspond to different timestamps, i.e., t_1, t_2,..., t_T [11].



(a) May 2020 (b) December 2020

(c) April 2021 (d) January 2022

Figure 2: Spatiotemporal hotspot maps for COVID-19 cases in Hong Kong.

Table 1 summarizes the main differences between LIBKDV and the existing software tools. Observe that LIBKDV is more efficient and versatile. In this paper, we first overview the technical details of LIBKDV in Section 2. Then, we discuss the functionality and benefits of LIBKDV in Section 3. Lastly, we discuss the demonstration plan in Section 4.

Table 1: Comparisons of different software tools.

✓ Natively support △ Partially support × Cannot support

Software tool	Time-complexity reduction	Versatility	
		Bandwidth-tuning analysis	Spatiotemporal analysis
QGIS	No	✓	×
ArcGIS	No	✓	×
KDV-Explorer	No	△	×
Deck.gl	No	△	×
Scipy	No	△	×
Statsmodels	No	△	×
Scikit-learn	No	△	×
LIBKDV (ours)	Yes	✓	✓

2 TECHNICAL OVERVIEW OF LIBKDV

In this section, we first formally define the problems in Section 2.1 that are related to the two analysis tasks. Then, we review the SLAM and SWS methods in Sections 2.2 and 2.3, respectively, which can reduce the time complexity for solving the problems in Section 2.1. After that, we discuss how to parallelize our methods, SLAM and SWS, to further improve the efficiency of LIBKDV in Section 2.4. Lastly, we discuss the technical novelty of LIBKDV against other software tools in Section 2.5.

2.1 Problem Definitions

We formally define KDV in Problem 1.

Problem 1. *Given a region with $X \times Y$ pixels and a set of n spatial data points $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$, we color each pixel $\mathbf{q}$ based on the kernel density function $\mathcal{F}_P(\mathbf{q})$ (cf. Equation 1), where b and $K^{(b)}_{space}(\mathbf{q}, \mathbf{p})$ denote the bandwidth and spatial kernel (cf. Table 2), respectively.*

$$\mathcal{F}_P(\mathbf{q}) = \frac{1}{n} \sum_{\mathbf{p} \in P} K^{(b)}_{space}(\mathbf{q}, \mathbf{p}) \tag{1}$$

Recall that domain experts need to perform bandwidth-tuning analysis, i.e., generate KDVs with multiple bandwidths and choose the most suitable one (cf. Figure 1). Therefore, we further define this bandwidth-tuning problem (cf. Problem 2).

Problem 2. *Given a set of L bandwidths, b_1, b_2,..., b_L, we need to generate KDV (i.e., solve Problem 1) for each bandwidth.*

Observe from Figure 2 that domain experts also need to obtain multiple hotspot maps with respect to different timestamps (or time-dependent KDVs). Here, we formally define this problem (cf. Problem 3).

Problem 3. *Given a set of T timestamps, i.e., $\mathcal{T} = \{t_1, t_2, ..., t_T\}$, a region with $X \times Y$ pixels, and a set of n spatiotemporal data points $\widehat{P} = \{(\mathbf{p}_1, t_{\mathbf{p}_1}), (\mathbf{p}_2, t_{\mathbf{p}_2}), ..., (\mathbf{p}_n, t_{\mathbf{p}_n})\}$, we adopt the spatiotemporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ (cf. Equation 2) to determine the color of each pixel $\mathbf{q}$ for each timestamp $t_i \in \mathcal{T}$, where b_s and b_t denote the bandwidths of spatial and temporal kernels, respectively.*

$$\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i) = \frac{1}{n} \sum_{(\mathbf{p}, t_\mathbf{p}) \in \widehat{P}} K^{(b_s)}_{space}(\mathbf{q}, \mathbf{p}) \cdot K^{(b_t)}_{time}(t_i, t_\mathbf{p}) \tag{2}$$

As a remark, the commonly-used spatial and temporal kernels are summarized in Table 2.

2.2 SLAM

In our preliminary work [13], we propose the state-of-the-art solution, called SLAM, to improve the efficiency for generating KDV (cf. Problem 1), which is briefly illustrated in Figure 3. Observe that this approach only takes $O(X + n)$ time to evaluate the kernel density function values $\mathcal{F}_P(\mathbf{q})$ for all pixels $\mathbf{q}$ in a row. Since there are Y rows in the region, the time complexity for using SLAM to generate KDV is $O(Y(X + n))$, which is significantly lower than the existing solutions (e.g., KDV-Explorer [12] with $O(XYn)$ time).

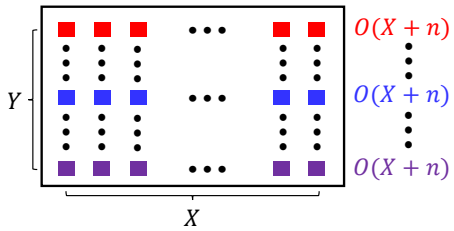


Figure 3: Brief illustration of SLAM, using the region with $X \times Y$ pixels. Computing the kernel density function values for all pixels with the same color (e.g., red) takes $O(X + n)$ time.

Compared with our previous work [13], we further extend SLAM to support the bandwidth-tuning problem (cf. Problem 2) in LIBKDV. Since SLAM only takes $O(Y(X + n))$ time to generate each KDV regardless of the bandwidth value, we can adopt SLAM to generate KDVs with multiple bandwidths, i.e., b_1, b_2,..., b_L, with $O(LY(X + n))$ time. Although this extension is simple, this approach can outperform the state-of-the-art method, SAFE [10], which takes $O(XY(L + n \log L))$ time (X is normally larger than L, e.g., $X = 640$ and $L = 20$ in [10]).

Table 2: Commonly-used spatial and temporal kernels, where $dist(\mathbf{q}, \mathbf{p})$ and $dist(t_i, t_\mathbf{p})$ denote the Euclidean distance functions.

Name	Spatial kernel ($K^{(b)}_{\text{space}}(\mathbf{q}, \mathbf{p})$)	Temporal kernel ($K^{(b)}_{\text{time}}(t_i, t_\mathbf{p})$)	Ref.
Epanechnikov	$\begin{cases} 1 - \frac{1}{b^2} dist(\mathbf{q}, \mathbf{p})^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 - \frac{1}{b^2} dist(t_i, t_\mathbf{p})^2 & \text{if } dist(t_i, t_\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases}$	[14, 15]
Quartic	$\begin{cases} (1 - \frac{1}{b^2} dist(\mathbf{q}, \mathbf{p})^2)^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} (1 - \frac{1}{b^2} dist(t_i, t_\mathbf{p})^2)^2 & \text{if } dist(t_i, t_\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases}$	[16, 22]

2.3 SWS

Unlike generating KDVs (cf. Problems 1 and 2), we cannot simply extend SLAM for supporting the spatiotemporal analysis (cf. Problem 3) as the spatiotemporal kernel density function $\mathcal{F}_P(\mathbf{q}, t_i)$ (cf. Equation 2) is more complex compared with the kernel density function (cf. Equation 1). Therefore, we adopt our preliminary work [11], called SWS, to handle this analysis task. Figure 4 briefly summarizes the idea of SWS. Observe that SWS can compute the spatiotemporal kernel density function values for all pixels of the same spatial position with T timestamps in $O(T + n)$ time. Since there are XY pixels for each visualization, SWS can generate T hotspot maps (cf. Figure 4) with $O(XY(T + n))$ time, which is better than the existing solutions [14–16] (with $O(XYTn)$ time).

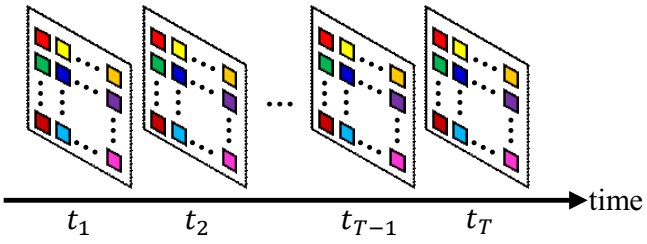


Figure 4: Brief illustration of SWS with T timestamps, using the region with $X \times Y$ pixels. Computing the spatiotemporal kernel density function values for all pixels with the same color (e.g., red) takes $O(T + n)$ time.

2.4 Parallelization of Our Methods

In this section, we extend our previous studies [11, 13] by considering how to parallelize our methods, SLAM and SWS, to further improve the efficiency of LIBKDV for solving Problems 1 to 3.

Parallelization of SLAM: In Figure 3, observe that we do not need to share the resources for computing different rows (e.g., row 1 and row 2) of pixels. Therefore, we adopt the round-robin approach to assign each thread to handle each row of pixels, which can fully parallelize our SLAM method.

Parallelization of SWS: Similarly, we adopt the round-robin approach to assign each thread for handling the same color pixels (with T timestamps) in Figure 4, e.g., thread 1 for the red color pixels and thread 2 for the yellow color pixels. Based on the same reason, we can also fully parallelize our SWS method.

2.5 Technical Novelty of LIBKDV

Recall that LIBKDV can reduce the worst-case time complexity for generating a single KDV (cf. Problem 1) and handling both the bandwidth-tuning analysis (cf. Problem 2) and spatiotemporal analysis (cf. Problem 3), which cannot be achieved by the existing tools. In the section, we elaborate more about the underlying reason for this issue.

Although many efficient methods have been incorporated into the off-the-self software tools to improve the efficiency for computing KDV, all these methods only exploit the optimization opportunity for the location data points, but not pixels. As an example, both Scikit-learn [18] and KDV-Explorer [12] incorporate the bound functions into a tree structure (e.g., kd-tree) to filter the data points. However, it is non-trivial to develop complexity-optimized algorithms for this approach since the location data points may not exhibit any property. Instead, LIBKDV is the first software tool to explore the regularity of pixels (e.g., all pixels in the same row have the same y-coordinate in Figure 3) to share computation, which can successfully reduce the worst-case time complexity for supporting KDV-based geospatial analysis.

3 FUNCTIONALITY AND BENEFITS OF LIBKDV

LIBKDV is a python package that is available online for supporting different types of KDV-based geospatial analysis tasks, including bandwidth-tuning analysis (cf. Figure 1) and spatiotemporal analysis (cf. Figure 2). Figure 5 shows how we call LIBKDV for generating the hotspot map in the Atlanta crime dataset, which follows these three steps.

Step 1 (Load dataset): We need to load the location dataset, which stores both the spatial coordinates and time in each row, into our library.

Step 2 (Compute): Users need to first initialize the parameters (e.g., choose the analysis task and set the number of pixels in the x-axis and y-axis). Then, LIBKDV adopts the corresponding algorithm in Section 2 to generate hotspot maps. More details for setting the parameters to support different geospatial analysis tasks can be found in our Github repository: https://github.com/libkdv/libkdv

Step 3 (Plot): We adopt Kepler.gl [5], a WebGL empowered high-performance web application, to plot the hotspot maps, which are based on the density values generated by our LIBKDV.

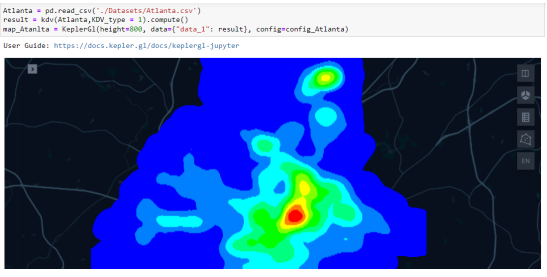


Figure 5: Using LIBKDV to generate the hotspot map in the Atlanta crime dataset.

Observe from Figure 5 that we only need to write three lines of python codes for using our LIBKDV to generate hotspot maps. Therefore, LIBKDV is an easy-to-use software package for which

the developer can easily integrate into their systems. Furthermore, since LIBKDV is more efficient compared with other python packages, e.g., Scikit-learn, users can call our functions instead of the KDV functions in these libraries.

4 DEMONSTRATION PLAN

In our demonstration, we use the Jupyter Notebook to show three key features of our LIBKDV, which are summarized as follows.

Efficiency of LIBKDV: To demonstrate that LIBKDV is the most efficient tool, we will first pre-install all the software tools (cf. Table 1) in a laptop. Then, we will prepare the python script which calls our LIBKDV and these tools for large-scale location datasets (e.g., New York traffic accident dataset [6]). Audience can notice the time gaps between LIBKDV and other tools, by running the python script.

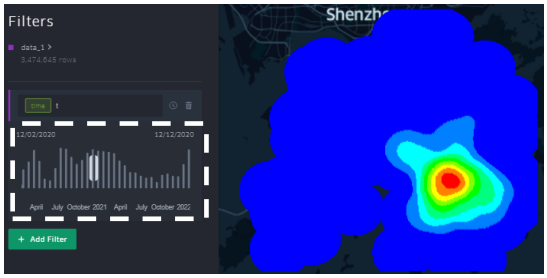


Figure 6: Using the sliding bar (in the dashed white box) to display multiple KDVs.

LIBKDV for bandwidth-tuning analysis: We will use four large-scale location datasets, namely Atlanta crime dataset [2], Seattle crime dataset [8], New York traffic accident dataset [6], and Hong Kong COVID-19 dataset [4], for demonstrating this feature. In our python script, we specify 5 to 20 bandwidths for each dataset and invoke Kepler.gl (cf. Figure 6) to display multiple KDVs. Audience can move the sliding bar (cf. the dashed white box in Figure 6) to visualize the changes of hotspot maps with respect to different bandwidths (cf. Figure 1) so that they can identify the most suitable one during the demonstration.

LIBKDV for spatiotemporal analysis: Like the demonstration for the bandwidth-tuning analysis, we adopt the same datasets and use the sliding bar (cf. Figure 6) to display multiple KDVs for this demonstration. By default, we choose 32 timestamps, i.e., $T = 32$, for each dataset (cf. Problem 3). In this demonstration, we show three case studies, which correspond to the crime hotspot detection, traffic accident hotspot detection, and COVID-19 hotspot detection. Audience can discover the hotspot patterns with respect to different timestamps (e.g., the spatial distribution and the waves of COVID-19 cases in Hong Kong).

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Plan of China (No.2019YFB2102100), the Science and Technology Development Fund Macau (SKL-IOTSC-2021-2023, 0015/2019/AKP), University of Macau (MYRG2019-00119-FST), IRCMS/19-20/H01, Hong Kong RGC Projects RIF R2002-20F, 12202221, and C2004-21GF. Pak Lon Ip, Kaiyan Zhao, and Leong Hou U are also affiliated with the State Key Laboratory of Internet of Things for Smart City.

REFERENCES

[1] [n. d.]. ArcGIS. http://pro.arcgis.com/en/pro-app/tool-reference/spatial-analyst/how-kernel-density-works.htm.
[2] [n. d.]. Atlanta Police Department Open Data. http://opendata.atlantapd.org/.
[3] [n. d.]. Deck.gl. https://deck.gl/docs/api-reference/aggregation-layers/heatmap-layer.
[4] [n. d.]. Hong Kong GeoData Store. https://geodata.gov.hk/gs/view-dataset?uuid=d4ccd9be-3bc0-449b-bd27-9eb9b615f2db&sidx=0.
[5] [n. d.]. Kepler.gl. https://kepler.gl/.
[6] [n. d.]. NYC Open Data. https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95.
[7] [n. d.]. QGIS. https://docs.qgis.org/2.18/en/docs/user_manual/plugins/plugins_heatmap.html.
[8] [n. d.]. Seattle Open Data. https://data.seattle.gov/Public-Safety/SPD-Crime-Data-2008-Present/tazs-3rd5.
[9] Tsz Nam Chan, Reynold Cheng, and Man Lung Yiu. 2020. QUAD: Quadratic-Bound-based Kernel Density Visualization. In *SIGMOD*. 35–50.
[10] Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. SAFE: A Share-and-Aggregate Bandwidth Exploration Framework for Kernel Density Visualization. *Proc. VLDB Endow.* 15, 3 (2022), 513–526.
[11] Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. SWS: A Complexity-Optimized Solution for Spatial-Temporal Kernel Density Visualization. *Proc. VLDB Endow.* 15, 4 (2022), 814–827.
[12] Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Weng Hou Tong, Shivansh Mittal, Ye Li, and Reynold Cheng. 2021. KDV-Explorer: A Near Real-Time Kernel Density Visualization System for Spatial Analysis. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2655–2658. http://www.vldb.org/pvldb/vol14/p2655-chan.pdf
[13] Tsz Nam Chan, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. SLAM: Efficient Sweep Line Algorithms for Kernel Density Visualization. In *SIGMOD*. 2120–2134.
[14] Eric Delmelle, Coline Dony, Irene Casas, Meijuan Jia, and Wenwu Tang. 2014. Visualizing the impact of space-time uncertainties on dengue fever patterns. *International Journal of Geographical Information Science* 28, 5 (2014), 1107–1127.
[15] Yujie Hu, Fahui Wang, Cecile Guin, and Haojie Zhu. 2018. A spatio-temporal kernel density estimation framework for predictive crime hotspot mapping and evaluation. *Applied Geography* 99 (2018), 89 – 97. https://doi.org/10.1016/j.apgeog.2018.08.001
[16] Jay Lee, Junfang Gong, and Shengwen Li. 2017. Exploring spatiotemporal clusters based on extended kernel estimation methods. *Int. J. Geogr. Inf. Sci.* 31, 6 (2017), 1154–1177. https://doi.org/10.1080/13658816.2017.1287371
[17] Yunxuan Li, Mohamed Abdel-Aty, Jinghui Yuan, Zeyang Cheng, and Jian Lu. 2020. Analyzing traffic violation behavior at urban intersections: A spatio-temporal kernel density estimation approach using automated enforcement system data. *Accident Analysis & Prevention* 141 (2020), 105509. https://doi.org/10.1016/j.aap.2020.105509
[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[19] Alina Ristea, Mohammad Al Boni, Bernd Resch, Matthew S. Gerber, and Michael Leitner. 2020. Spatial crime distribution and prediction for sporting events using social media. *Int. J. Geogr. Inf. Sci.* 34, 9 (2020), 1708–1739. https://doi.org/10.1080/13658816.2020.1719495
[20] Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.
[21] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2
[22] Kun Xie, Kaan Ozbay, Abdullah Kurkcu, and Hong Yang. 2017. Analysis of Traffic Crashes Involving Pedestrians Using Big Data: Investigation of Contributing Factors and Identification of Hotspots. *Risk Analysis* 37, 8 (2017), 1459–1476. https://EconPapers.repec.org/RePEc:wly:riskan:v:37:y:2017:i:8:p:1459-1476
[23] Suyan Yi, Hongwei Wang, Shengtian Yang, Ling Xie, Yibo Gao, and Chen Ma. 2021. Spatial and Temporal Characteristics of Hand-Foot-and-Mouth Disease and Its Response to Climate Factors in the Ili River Valley Region of China. *International Journal of Environmental Research and Public Health* 18, 4 (2021).
[24] Hao Yu, Pan Liu, Jun Chen, and Hao Wang. 2014. Comparative analysis of the spatial analysis methods for hotspot identification. *Accident Analysis and Prevention* 66 (2014), 80 – 88. https://doi.org/10.1016/j.aap.2014.01.017