

Visualizing Socioeconomic Data in India: A Comprehensive Methodology for Choropleth Map Creation

Vibhuti Mukesh Dhande

Electronics & Telecommunication
MKSSS's Cummins College of
Engineering for Women
Pune, India

vibhuti.dhande@cumminscollege.in

Pradnya Rajeev Kanale

Electronics & Telecommunication
MKSSS's Cummins College of
Engineering for Women
Pune, India

pradnya.kanale@cumminscollege.in

Maithili Madhav Karlekar

Electronics & Telecommunication
MKSSS's Cummins College of
Engineering for Women
Pune, India

maithili.karlekar@cumminscollege.in

Kashmira Kirtikumar Lodha

Electronics & Telecommunication
MKSSS's Cummins College of
Engineering for Women
Pune, India

kashmira.lodha@cumminscollege.in

Dr. Anita Patil (Professor)

Electronics & Telecommunication
MKSSS's Cummins College of Engineering for Women
Pune, India

anita.patil@cumminscollege.in

DOI: 10.26821/IJSHRE.11.5.2023.110509

ABSTRACT

Choropleth maps are a valuable tool for visualizing spatial patterns in socioeconomic data. In this paper, a methodology is presented for creating choropleth maps and demonstrate its implementation using the Socioeconomic High-resolution Rural-Urban Geographic Platform for India (SHRUG) dataset. This methodology comprises three main stages: data selection and preparation, data processing, and map creation. It emphasizes the value of documentation and reproducibility and offers thorough explanations and step-by-step procedure for each stage. Examples of choropleth maps produced using SHRUG data, encompassing a variety of socioeconomic variables at the village and town levels are presented to illustrate the utility of this technology. The example shows the possibilities of using SHRUG to look at socio economic problems in India and the effectiveness of choropleth maps in exposing geographical patterns and trends in complex data. The examples provided here serve as

a starting point for researchers interested in studying the rich data available through SHRUG and the methodology can be adapted to different datasets and contexts. In order to choose visualizations that may be utilized in a variety of fields, including the social sciences, economics, and public health, where choropleth maps are frequently used for data visualization, the research addresses the difficulties of working with big data sets. The recommendations for improving the usability of SHRUG data, including documentation, metadata, and software tools for data processing and visualization are also presented. Overall, this work emphasizes the significance of open-access data sources like SHRUG and offers a comprehensive and accessible guide for creating choropleth maps, with specific examples utilizing SHRUG data.

Keywords: SHRUG, choropleth, EDA, geospatial visualization.

1. INTRODUCTION

Choropleth can be a prevalent strategy to visualize spatial data on an outline, where locales are shaded or coloured based on the size of a variable of interest. They are powerful tools for visualizing spatial patterns in socioeconomic data to convey complex information and patterns across geographic areas, but creating them can be a challenging process. They are effective devices for visualizing spatial designs in financial information to communicate complex data and designs over geographic regions, but making them can be a challenging process. One of the biggest challenges is managing huge and complex datasets that require cautious determination and preparing to create significant maps, inconsistent geographic areas, information quality issues, and the ought to limit down the dataset to factors can moreover posture noteworthy challenges. In order to address these issues, we propose a data processing step that includes data fusion, dimensionality reduction, and data transformation techniques to reduce data volume and ensure meaningful map visualization.

Our methodology consists of three main stages: data selection and preparation, data processing, and map creation. The data selection and preparation stage carefully select and pre-processes the data to ensure that the data are suitable for generating choropleth mapping. In the data processing stage, statistical techniques are applied to transform and aggregate the data to address issues such as missing data and inconsistent geographic locations. Finally, in the map creation stage, we have used the processed data to create the choropleth maps that reveal spatial patterns and trends in the data.

This methodology involves using Python and its libraries. We recommend using Python for its ease of use, flexibility, and rich set of data analysis and visualization tools. Popular Python libraries such as pandas, geopandas, matplotlib, and plotly are used for data processing and visualization.

To demonstrate the applicability of our methodology, we provide several examples of choropleth maps created using SHRUG data, covering a range of socioeconomic indicators at the village and town levels. These examples illustrate the power of choropleth maps in revealing spatial patterns and trends in complex data and showcase the potential for using SHRUG to investigate socioeconomic issues in India. Our methodology is applicable to other datasets and contexts, and our example serves as a starting point for researchers interested in exploring the wealth of data available through SHRUG.

Performing exploratory data analysis (EDA) is a crucial component of any research analysis. The primary objective of conducting exploratory

analysis is to scrutinize the data for any anomalies, outliers, and distribution patterns to guide targeted testing of the hypothesis. This technique also offers useful tools to generate hypotheses by visualizing and comprehending the data through graphical representation [1]. Furthermore, EDA often includes techniques for selecting relevant features. However, many sources explaining different EDA techniques fail to highlight the importance of visualizing the geospatial data through visualizations like choropleths, heatmaps, cartograms, point maps, proportional symbol maps, etc.

2. LITERATURE REVIEW

The paper ‘*The Socioeconomic High-resolution Rural-Urban Geographic Dataset on India (SHRUG)*’ [2] by Asher. S, Lunt. T, Matsuur. R. and Novosad. P, documents the usage suggestions and applications of the SHRUG dataset based in India. What makes the SHRUG platform unique is the framework of the collected data. The lack of a unified platform to interlink these datasets results in significant replication of effort, while also leading to missed opportunities for possible synergies between projects. Researchers studying geographical variations in India can reap advantages by connecting their data with the SHRUG platform. In turn, they can also provide benefits to other researchers by sharing their data through the same platform.

The paper ‘*An Evaluation of Visualization Methods for Population Statistics Based on Choropleth Maps*’ [3] by Besançon, Lonni, et al. describes the usage of the colour mapping symbology technique in-depth and also suggests suggestions for the challenges faced while implementing them. The study found that choropleth maps can be misleading for viewers when large regions with low population densities dominate the map. The data selection and preparation stage carefully select and pre-processes the data to ensure that the data are suitable for generating choropleth mapping. The authors highlight the potential of 3D choropleth maps and point out the low accuracy of choropleth map tasks using multivariate data. Additionally, the study presents and evaluates four of his techniques, called pop charts. Pop charts are intended to represent population density at finer scales on a choropleth map.

3. METHODOLOGY

Using this methodology, programmers can create high-quality choropleth maps that precisely represent the data they are trying to visualize, making it simpler to form conclusions and make decisions based on the insights gained from the

data. Furthermore, a working knowledge of choropleth map methodology might lead to new opportunities for programmers in disciplines like geography, urban planning, and public policy, where the capacity to effectively visualize and analyze geographic data is crucial.

A. Dataset selection

In order to create a choropleth map, both geographical and statistical data is needed. Data is imported and converted into a data frame. Having the data in a data frame makes it easier to manipulate and work with the data, allowing the programmer to create accurate and informative choropleth maps.

When using Python, the *'pandas'* library makes it easier to work with the data frame rather than the raw file. It supports 14 different file formats that can be converted into a data frame.

B. Operations on the data frame

The column to be selected needs to be mapped to the data frame. Then, filter out the required column along with the associated geographical data column, and copy them into a new data frame. It is not recommended to drop columns from the original data frame. This process allows programmers to work with a smaller, more manageable data frame and avoid accidentally deleting necessary data.

C. Importance of keys

When working with spatial data, the area being described is likely to be the primary key in the dataset. This means that the most unique parameter for each row of data is the area it represents. Many datasets will have an index representing the names of the locations being analysed. However, if there is no such column directly available, there will likely be a separate key dataset supporting the primary dataset. To incorporate this key data, read the data and convert it into a data frame, then merge it with the original data frame on a common index column. Finally, filter out the relevant columns into a new data frame. This process ensures that a unified and accurate dataset is prepared to work with.

D. Granularity

When creating a choropleth map, it's important to define the level of granularity required for the analysis. Depending on the data, the map can be plotted country-wise, state-wise, district-wise, sub-district-wise, or village/town-wise. To achieve the desired granularity, grouping functions of the

chosen programming language on the geographical parameter column can be used.

It is also essential to choose the appropriate aggregation function for the data column. While several aggregation operations are available, the most commonly used are sum and mean. To determine the best aggregation function to use, consider the type of data being plotted. It makes sense, to sum up, total population numbers, but it doesn't make sense, to sum up, literacy rates. Taking mean is a better operation for combining multiple percentages. By using the appropriate aggregation function, the accuracy of the choropleth map can be ensured.

E. Base map

After extracting the required data for the choropleth map, the next step is to obtain an actual map to plot it onto. Two formats are commonly used for plotting choropleths: shapefile and geojson. Both formats contain geometries that provide the latitude, longitude, and ID information for the locations. Shapefiles are straightforward to read using geopandas. In contrast, in geojson, we need to extract and store the geometry and location names from the features array and store them in a separate dictionary. It's essential to ensure that the granularity of the shapefiles/geojson files is the same as that of the primary data frame. All the information contained in the shapefiles/geojson files can be seen by simply printing them after importing them.

F. Operations on the geometric information

For accurate mapping, the geographical location information in the shapefile or geojson file and primary data frame must match exactly. The column is checked for strict equality during mapping, making case sensitivity an important consideration. Anomalies can be corrected by altering the case of letters or dropping words to ensure string equality. To extract the necessary data from the shapefile, two columns are required: the geometry column and the index column, which must be common with the primary data frame.

G. Merging geometric information with the data frame

Merge the shapefile and our data frame on the common column so that the entire information is in one single data frame: the geometry, the data to be mapped, as well as the geographical data.

The Levenshtein distance algorithm can be used to determine the best match between location names in the shapefile/geojson and data frame, based on the desired level of granularity. The Levenshtein distance algorithm calculates the minimum number of single-character edits

(insertions, deletions, or substitutions) required to transform one string into another, to find the closest match between the two datasets.

H. Plotting

If Python is used to generate choropleth maps, matplotlib's plot() function by specifying the column to be mapped and adding a legend or custom color scale can be used. Alternatively, Plotly has an inbuilt choropleth() function that takes multiple arguments to create interactive and flexible maps. Additionally, there are other functions available in R programming language and MATLAB for generating choropleth maps.

4. DISCUSSIONS AND RESULTS

Plotting a choropleth map for the SHRUG dataset using Python programming language.

A. Dataset selection

Considering 'Population Census dataset 2011' from the SHRUG platform. The downloaded file from the website is a CSV file. It is fairly easy to work with the Python library 'pandas' data frame instead of working with the raw file.

The CSV version of the population census 2011 is pulled up into a pandas data frame 'population_data_11'.

Every dataset has a primary key. Here 'shrid' corresponds to the unique ID for every single town if the data is urban and the village if the data is rural.

	shrid	pc11_pca_tot_p	pc11_pca_tot_p_r	pc11_pca_tot_p_u	pc11_pca_no_hh	pc11_pca_p_sc	pc11_pca_p_st	pc11_pca_p_ll
0	11-01-000001	130.0	130.0	NaN	130	0	0	130.0
1	11-01-000002	3770.0	3770.0	NaN	400	0	0	2580.0
2	11-01-000005	5255.0	5255.0	NaN	528	0	0	2140.0
3	11-01-000006	3001.0	3001.0	NaN	354	0	349	1364.0
4	11-01-000007	1310.0	1310.0	NaN	280	0	302	208.0
...
590869	11-35-645567	183.0	183.0	NaN	41	0	0	132.0
590870	11-35-645568	147.0	147.0	NaN	38	0	0	103.0
590871	11-35-645569	183.0	183.0	NaN	46	0	0	136.0
590872	11-35-645570	67.0	67.0	NaN	23	0	0	56.0
590873	11-35-804041	108058.0	NaN	108058.0	27049	0	1476	87694.0

590874 rows x 9 columns

Fig.1 Population dataset

B. Operations on Data frame

To determine the column to plot on the choropleth map, we consider the available columns in the data frame. In this particular case, there are 32 columns, including **pc11_pca_p_lit** which

represents the total literate population and **pc11_pca_tot_p** which represents the total population. To compute the literacy rate, a lambda function can be used by dividing the literate population by the total population, resulting in a new column called **pc11_pca_p_lit_rate**.

```

In [ ]: import pandas as pd

In [ ]: population_data_11 = pd.read_csv("shrug-v1.5.samcra-pop-econ-census-csv/shrug_pc11.csv")
        population_data_11['pc11_pca_p_lit_rate'] = population_data_11.apply(lambda x: ((x['pc11_pca_p_lit']/x['pc11_pca_tot_p'])*100)
        population_data_11

Out[ ]:
pc11_vd_power_agr  pc11_vd_power_all  pc11_hd_p_sch  pc11_hd_m_sch  pc11_hd_s_sch  pc11_hd_college  pc11_hd_s_s_sch  pc11_hd_area  pc11_pca_p_lit_rate
0  0.0  0.0  NaN  NaN  NaN  NaN  NaN  NaN  100.000000
1  0.0  1.0  NaN  NaN  NaN  NaN  NaN  NaN  68.435013
2  0.0  1.0  NaN  NaN  NaN  NaN  NaN  NaN  40.894386
3  0.0  1.0  NaN  NaN  NaN  NaN  NaN  NaN  45.451516
4  0.0  1.0  NaN  NaN  NaN  NaN  NaN  NaN  15.877863
...  ...  ...  ...  ...  ...  ...  ...  ...
590869  ...  ...  ...  ...  ...  ...  ...  ...  72.131148
590870  ...  ...  ...  ...  ...  ...  ...  ...  70.068027
590871  ...  ...  ...  ...  ...  ...  ...  ...  74.316940
590872  ...  ...  ...  ...  ...  ...  ...  ...  83.582090
590873  ...  ...  ...  ...  ...  ...  ...  ...  81.339651
    
```

Fig.2 Newly created literacy rate column

Two columns are needed for creating a choropleth map: statistical data and a geographic key. Here, **pc11_pca_p_lit_rate** is the data column and **shrid** is the geography column. Remove the unnecessary columns.

```

In [ ]: #filtering out the data and geography column
        population_data_lit = population_data_11[['pc11_pca_p_lit_rate', 'shrid']]
        population_data_lit

Out[ ]:
pc11_pca_p_lit_rate  shrid
0  100.000000  11-01-000001
1  68.435013  11-01-000002
2  40.894386  11-01-000005
3  45.451516  11-01-000006
4  15.877863  11-01-000007
...  ...  ...
590869  72.131148  11-35-645567
590870  70.068027  11-35-645568
590871  74.316940  11-35-645569
590872  83.582090  11-35-645570
590873  81.339651  11-35-804041
    
```

590874 rows x 2 columns

Fig.3 After the removal of unnecessary columns

The relevant information is filtered out in a different data frame **population_data_lit**.

C. Importance of keys

This dataset has a primary key as 'shrid' and it only contains numerical information. It does not exactly mention the state, district, or place name.

SHRUG platform has provided the geographical names associated with the 'shrid'.

Merge these two data frames, keys, and *population_data_lit*. This will result in a data frame that contains the state, district, and subdistrict names in the same data frame as the parameters.

```
#merging keys df with the main df
population_data_lit = population_data_lit.merge(keys,on='shrid')
population_data_lit
```

	pc11_pca_p_lit_rate	shrid	state_name	district_name	subdistrict_name	place_na
0	100.000000	11-01-000001	jammu kashmir	kupwara	kupwara	B
1	68.435013	11-01-000002	jammu kashmir	kupwara	kupwara	Ke
2	40.894386	11-01-000005	jammu kashmir	kupwara	kupwara	Mindt
3	45.451516	11-01-000006	jammu kashmir	kupwara	kupwara	Pa
4	15.877863	11-01-000007	jammu kashmir	kupwara	kupwara	Juma Gund

```
590874 rows x 6 columns
```

Fig.4 key-value pair

After that drop null values and filter out the important columns.

```
#cleaning and filtering
population_data_lit.dropna(inplace=True)
population_data_lit = population_data_lit[['pc11_pca_p_lit_rate', 'state_name']]
population_data_lit
```

	pc11_pca_p_lit_rate	state_name
0	100.000000	jammu kashmir
1	68.435013	jammu kashmir
2	40.894386	jammu kashmir
3	45.451516	jammu kashmir
4	15.877863	jammu kashmir

```
590868 rows x 2 columns
```

Fig.5 After merging and filtering the columns

D. Granularity

For this example, consider 'statewise' granularity. Data frame column '*pc11_pca_p_lit_rate*' defines the literacy rate for each town/village in every state. It makes sense to take the mean() of this data for every state.

```
#grouping according to states
population_data_lit = population_data_lit.groupby(by = 'state_name',as_index=False).mean()
population_data_lit
```

	state_name	pc11_pca_p_lit_rate
0	andaman nicobar islands	68.278926
1	andhra pradesh	49.147142
2	arunachal pradesh	45.770795
3	assam	59.478303
4	bihar	49.369819
5	chandigarh	76.312284
6	chhattisgarh	53.447766
7	dadra nagar haveli	53.187349
8	daman diu	75.838582
9	goa	77.551261
10	gujarat	60.457085
11	haryana	62.582589
12	himachal pradesh	72.287649
13	jammu kashmir	54.396281
14	jharkhand	48.268790
15	karnataka	61.567070
16	kerala	83.500547

Fig.6 Data frame after 'mean' operation

E. Base Map

The shapefile includes state code in 'shrid', state name, and geometry data for 34 Indian states and union territories, making it possible to map data for any state. However, the shapefile does not have 'shrid' as the ID, but it provides state names directly. Therefore, merge the keys with our data frame to maintain consistency in columns.

```
shapefile= r"shapefiles\state.shp"
state_shp = gpd.read_file(shapefile)
```

	pc11_s_id	s_name	geometry
0	01	Jammu and Kashmir	POLYGON ((77.95837 35.48178, 77.96405 35.48433...
1	02	Himachal Pradesh	POLYGON ((76.80943 33.23872, 76.81593 33.23535...
2	03	Punjab	POLYGON ((75.83876 32.52156, 75.83898 32.52128...
3	04	Chandigarh	POLYGON ((76.79191 30.77115, 76.79229 30.77118...
4	05	Uttarakhand	POLYGON ((79.22439 31.34099, 79.22624 31.33888...
5	06	Haryana	POLYGON ((76.84307 30.88633, 76.84365 30.88618...
6	07	NCT Of Delhi	POLYGON ((77.07688 28.88184, 77.07801 28.88149...
7	08	Rajasthan	POLYGON ((73.90898 30.05334, 73.90437 30.05000...
8	09	Uttar Pradesh	MULTIPOLYGON (((79.36095 25.13890, 79.36217 25...
9	10	Bihar	MULTIPOLYGON (((84.51370 24.25774, 84.51182 24...
10	11	Sikkim	POLYGON ((88.65147 28.09304, 88.65270 28.09111...
11	12	Arunachal Pradesh	POLYGON ((96.11495 29.39999, 96.14150 29.36881...
12	13	Nagaland	POLYGON ((95.19597 26.99497, 95.19551 26.99432...
13	14	Manipur	POLYGON ((94.57290 25.69122, 94.57351 25.68996...
14	15	Mizoram	POLYGON ((92.76760 24.51982, 92.76935 24.51554...
15	16	Tripura	POLYGON ((92.17048 24.53087, 92.17477 24.52727...

Fig.7 Shapefile

F. Operations on the geometric information

To ensure accurate mapping, data values must match exactly, including cases. Any discrepancies due to different naming conventions changes over time, or errors must be corrected. Next, extract two columns from the shapefile: the geometry column

and the index column with a common state name in our primary data frame.

	geometry	state_name
0	POLYGON ((77.95837 35.48178, 77.96405 35.48433...	jammu kashmir
1	POLYGON ((76.80943 33.23872, 76.81593 33.23535...	himachal pradesh
2	POLYGON ((75.83876 32.52156, 75.83898 32.52128...	punjab
3	POLYGON ((75.79191 30.77115, 76.79229 30.77118...	chandigarh
4	POLYGON ((79.22439 31.34099, 79.22624 31.33888...	uttarakhand
5	POLYGON ((76.84307 30.88633, 76.84365 30.88618...	haryana
6	POLYGON ((77.07688 28.88184, 77.07801 28.88149...	nct of delhi
7	POLYGON ((73.90898 30.05334, 73.90437 30.05000...	rajasthan
8	MULTIPOLYGON (((79.36095 25.13890, 79.36217 25...	uttar pradesh
9	MULTIPOLYGON (((84.51370 24.25774, 84.51182 24...	bihar
10	POLYGON ((88.65147 28.09304, 88.65270 28.09111...	sikkim
11	POLYGON ((96.11495 29.39999, 96.14150 29.36881...	arunachal pradesh
12	POLYGON ((95.19597 26.99497, 95.19551 26.99432...	nagaland
13	POLYGON ((94.57290 25.69122, 94.57351 25.68996...	manipur
14	POLYGON ((92.76760 24.51982, 92.76935 24.51554...	mizoram
15	POLYGON ((92.17048 24.53087, 92.17477 24.52727...	tripura
16	POLYGON ((91.87533 26.08687, 91.87467 26.08240...	meghalaya

Fig.8 Shapefile with geometry and state names

G. Merging geometric information with the data frame

To ensure accurate mapping of data on the choropleth map, merge the *keys* dataset and the *population_data_lit* dataset. The resulting merged dataset contains state, district, and subdistrict names along with the corresponding parameters.

```

M #merging keys df with the main df
population_data_lit = population_data_lit.merge(keys, on='shrid')
population_data_lit

In [ ]:

```

	pct11_pca_p_lr_rate	shrid	state_name	district_name	subdistrict_name	place_name
0	100.000000	11-01-000001	jammu kashmir	kupwara	kupwara	Bore
1	68.435013	11-01-000002	jammu kashmir	kupwara	kupwara	Keran
2	40.894386	11-01-000005	jammu kashmir	kupwara	kupwara	Mindyan
3	45.451516	11-01-000006	jammu kashmir	kupwara	kupwara	Patrin
4	15.877863	11-01-000007	jammu kashmir	kupwara	kupwara	Juma Gund
...
590869	72.131148	11-35-645567	andaman nicobar islands	south andaman	little andaman	Butler Bay Forest Camp 4-IV (FDCA)
590870	70.068027	11-35-645568	andaman nicobar islands	south andaman	little andaman	Red Oil Palm (Nursery Camp)
590871	74.316940	11-35-645569	andaman nicobar islands	south andaman	little andaman	Butler Bay Forest Camp 4-II (FDCA)
590872	83.582090	11-35-645570	andaman nicobar islands	south andaman	little andaman	Butler Bay Forest Camp 4-I (FDCA)
590873	81.339651	11-35-804041	andaman nicobar islands	south andaman	port blair	port blair

590874 rows x 6 columns

Fig.9 After merging data frame with the shapefile

H. Plotting

Here the 'matplotlib' or 'plotly' library of Python can be used to plot the choropleth map of the data frame.

Plotly maps are more interactive, users can hover as well as adjust a lot of other parameters such as the number of columns to be displayed on hover, users can filter out the exact location of the map, hide the rest of the map, etc.

It has an inbuilt function 'plotly.express.choropleth()' that takes in multiple arguments.

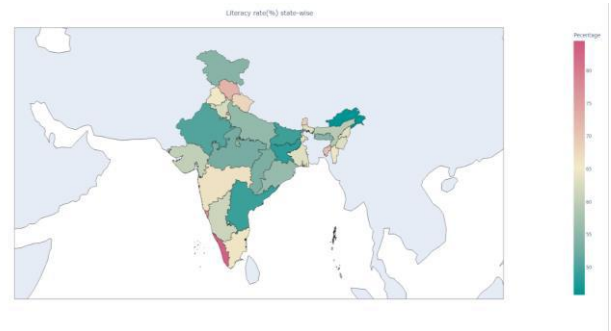


Fig.9 Final output: Choropleth map of India for literacy rate

5. CONCLUSION

Choropleth and other visualizations greatly improve our understanding of complex datasets such as the

SHRUG dataset. By visualizing data in spatial context, you can identify patterns and trends that aren't readily apparent when looking at data in tabular form. Choropleth, in particular, is a powerful visualization tool that helps you visualize data at a regional or local level, making it easier to see inequalities and differences between different regions.

However, it is important to note that choosing the appropriate visualization technique for a dataset depends on various factors, such as the type of data, the research question, and the target audience. By choosing appropriate visualization techniques, researchers and policymakers can gain a deeper understanding of the factors that impact socioeconomic development in different regions and identify areas where intervention is needed to promote more equitable development across the country.

We highlight the importance of data-driven research and encourage the use of open-access data repositories like SHRUG for advancing our understanding of complex socioeconomic systems. By providing a comprehensive and accessible guide for creating choropleth maps, we hope to empower researchers to use this powerful visualization tool to explore spatial patterns and trends in socioeconomic data and contribute to the growing field of data-driven research in India and beyond.

6. ACKNOWLEDGEMENT

We express our gratitude towards Dr. Shirrang Karandikar (AlgoAsylum) for his guidance and support. Lastly, we extend our thanks to everyone who has supported us throughout this project.

7. REFERENCES

1. Data, MIT Critical, Matthieu Komorowski, Dominic C. Marshall, Justin D.

- Salciccioli, and Yves Crutain. "Exploratory data analysis." *Secondary analysis of electronic health records*: 185-203 (2016).
2. Asher, Sam, Tobias Lunt, Ryu Matsuura, and Paul Novosad. "The socioeconomic high-resolution rural-urban geographic dataset on India (SHRUG)." URL: <https://doi.org/10.7910/DVN/DPESAK> (2019).
3. Besançon, Lonni, Matthew Cooper, Anders Ynnerman, and Frédéric Vernier. "An evaluation of visualization methods for population statistics based on choropleth maps." arXiv preprint arXiv:2005.00324 (2020).
4. Kumar, Hemanshu, and Rohini Somanathan. "State and district boundary changes in India: 1961-2001." Available at SSRN 2687484 (2015).
5. Leonowicz, Anna. "Two-variable choropleth maps as a useful tool for visualization of geographical relationship." *Geografija* 42: 33-37 (2006).
6. Government of India (1996) *General Population Tables and Primary Census Abstract Census of India 1991: Series 8: Haryana* (Chandigarh: Director of Census Operations, Haryana) (2016)
7. Das, Sanjeev Kumar, and Sujit Beborrtta. "A study on geospatially assessing the impact of COVID-19 in Maharashtra, India." *The Egyptian Journal of Remote Sensing and Space Science* 25, no. 1: 221-232 (2022).
8. Post, Frits H., Gregory Nielson, and Georges-Pierre Bonneau, eds. "Data visualization: The state of the art." (2002).