C19EpidemiologyGraphs: A Spatial Graph Database with Global COVID-19 Epidemiological Vertex Features

Artur B. Torres, Gladston J. P. Moreira[1]

[1]Computing Department – Universidade Federal de Ouro Preto (UFOP)
Ouro Preto, MG – Brazil

arturtorres@proton.me, gladston@ufop.edu.br

Abstract. *Adding spatial relationships to a collection of independently reported pieces of data can be a challenging prerequisite before using spatial-aware tools, such as graph neural networks. At the same time, epidemiological research could benefit from the spatial awareness these tools provide while exploring new epidemic models. With this in mind, we introduce C19EpidemiologyGraphs, a global and multi-aggregation level spatial graph database embedded with vertex-level features related to COVID-19 epidemiology, where vertices are reporting locations and edges indicate that two locations share a border. To the best of our knowledge, this is the first public database to integrate spatial graphs with COVID-19 epidemiological data at a large scale.*

1. Introduction

Epidemiological data often plays a key role in decision-making during public health emergencies, as it can be used by researchers and analysts to, among other things, better understand how certain diseases spread and the impact some government policies have on fighting them. However, this data is often made available by each reporting agency individually, obscuring spatial relationships between different locations. For instance, if a certain city has an outbreak, does this mean anything for neighboring cities? What about the neighbors' neighbors?

Introducing spatial relations may aid researchers in identifying outbreak hotspots and in creating more robust models for time-series analyses by accounting not only for a specific location's epidemiological history but also for the histories of nearby places. With this possibility in mind, this work introduces C19EpidemiologyGraphs, an SQLite database containing COVID-19 epidemiological data organized into graphs, where each vertex represents a location, and each edge signals that two locations share a border. The database encompasses three levels of aggregation: one for countries, another for intermediate administrative divisions (such as states, districts, or prefectures, among others) and a third for municipalities (such as cities or towns). These will be referred to as levels 0, 1, and 2, respectively.

We utilize Google's COVID-19 Open Dataset[1] (COD) [Wahltinez et al. 2022] for the epidemiological information and GADM[2] version 4.1 for the geometric shapes of administrative areas, the latter being used to create the graph edges based on whether the shapes of two locations intersect or not. For C19EpidemiologyGraphs, we adopt COD's

[1]Available at: https://health.google.com/covid-19/open-data/
[2]Available at: https://gadm.org

terminology of referring to administrative division levels as *aggregation levels*, or just *levels*. GADM uses the word *layers* to describe its own subdivisions, such that layer 0 is for countries, layer 1 is for first-level administrative subdivisions, layer 2 is for second-level, and so on.

Both the database files and the source code to create them are publicly available at this project's GitHub page[3]. The source code is licensed under BSD-3-Clause and the processed database file is dual-licensed in the following way: the database's schema and most of the data contained within it are under CC BY 4.0, with the only exception being the graph edges that have one or both of their ends connected to an Austrian location of any level of aggregation, as those are instead under CC BY-SA 4.0 in order to conform with the GADM license.

By assembling a graph database, this work aims to facilitate the usage of emerging technologies that operate on this kind of data, notably graph neural networks (GNNs), which have been studied for their potential in some epidemiological tasks, such as epidemic prediction [Liu et al. 2024]. We hope that this project's contributions will assist in the development of more robust statistical models that might, in turn, aid in meaningful decision-making by authoritative agencies.

2. Related Works

Several works that utilize graph neural networks in an epidemiological context have used small-scale datasets consisting of only a few unique locations. The work defining Cola-GNN [Deng et al. 2020] creates adjacency matrices for Japanese prefectures, American regions, and American states, and then associates these locations with epidemiological data on Influenza-like illnesses, with the goal of studying outbreak forecasting. The datasets assembled in Cola-GNN's paper are also used in the work that defines EpiGNN [Xie et al. 2022], which also constructs graphs for Australian states and territories and Spanish NUTS3 subdivisions, this time associating them with COVID-19 epidemiological features.

COVID-19 Data Hub [Guidotti and Ardia 2020] [Guidotti 2022] offers a more global alternative: it's a data aggregation tool that combines epidemiological data with GADM indices for spatial analyses. The provided data is not organized into graphs, but as each location has, whenever possible, a key to a GADM database entry, which is the same spatial source used by us, it could be possible to create graphs by employing a method similar to the one used to create C19EpidemiologyGraphs. However, using COVID-19 Data Hub's GADM association would yield different results than what is achieved in C19EpidemiologyGraphs, as our association of the spatial data source with the epidemiological one was done with an added one-to-one constraint and some manual work that differs from the techniques employed in COVID-19 Data Hub.

It may be worth noting that C19EpidemiologyGraphs is not meant to be a data aggregation tool, such as COVID-19 Data Hub, and it actually relies on one of such tools to achieve its objectives. C19EpidemiologyGraphs' main goal is to build global graphs at multiple levels of aggregation and integrate them with an external epidemiological data source, striving to facilitate the creation of large and custom epidemiological datasets that are suitable for graph-based statistical processing.

[3]Available at: https://github.com/bermondd/C19EpidemiologyGraphs

To the best of our knowledge, no other work has provided a global and multi-level database of graphs with associated COVID-19 epidemiological vertex features.

3. Building the database

C19EpidemiologyGraphs is available as an SQLite database with three tables:

- **vertices**: Defines the graph nodes as locations present in both COD and GADM. All columns are taken from COD, with the exception of GADM_layer and GADM_index, which inform, respectively, what GADM layer and what row index in that layer this vertex is representing. This table can model both GADM and COD entries because these sources are forced to be joined with one another via a one-to-one cardinality. This join process is described in Section 3.1.
- **edges**: Defines graph edges by referencing their vertices' location keys. This table also includes a distance measurement between the source and destination nodes, calculated as the geodesic distance, in meters and on the WGS 84 Earth surface, between the coordinates provided for both locations in the vertices table. The method for calculating edges using GADM shapes is described in Section 3.2.
- **epidemiology**: Defines COVID-19 epidemiological snapshots for a given location and at a given date by reproducing the data available in COD. These snapshots can then be used to assign features to the vertices and/or filter them, creating graphs with node attributes. This table's creation process is described in Section 3.3.

More details about the tables can be seen in Figure 1, where PK stands for Primary Key, FK for Foreign Key and NN for Not Null.

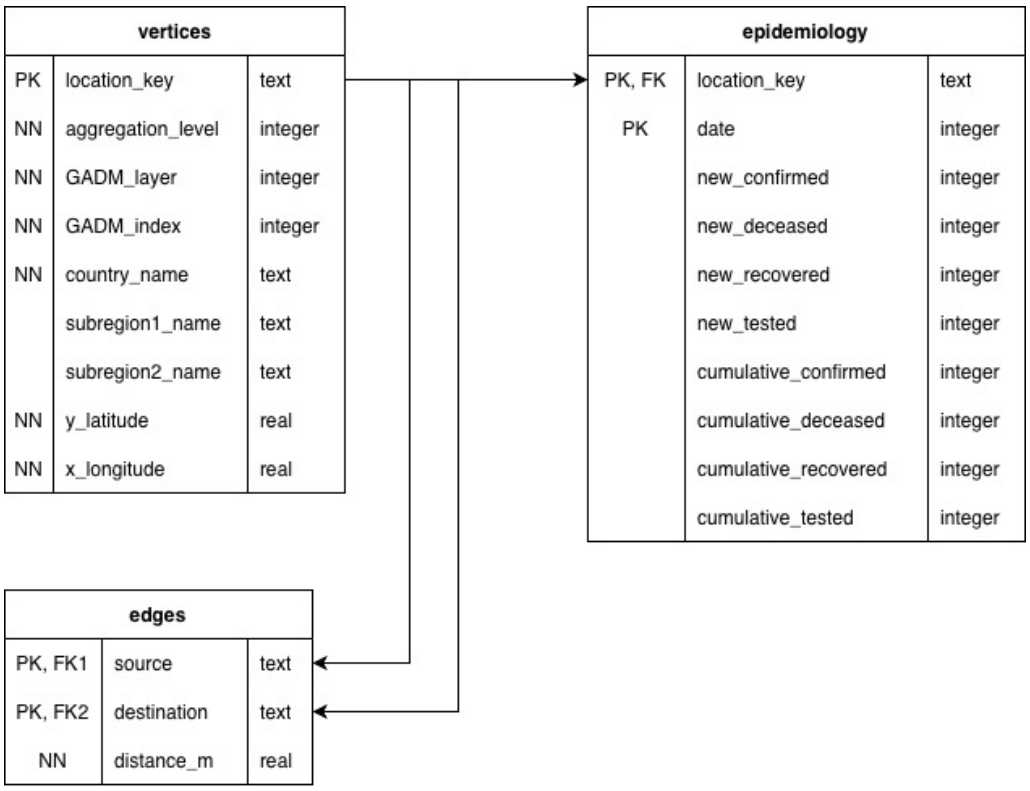


Figure 1. C19EpidemiologyGraphs' database schema

3.1. Relating the data sources

In order to build C19EpidemiologyGraphs, it's necessary to create connections between GADM and COD. As the former is used to create the graph edges and the latter to assign features to the vertices, they must be able to complement each other in C19EpidemiologyGraphs. They are two distinct data sources with different ways to label their locations, so it's necessary to create a way to join one with the other.

Before describing how this join is made, an important detail is the enforced cardinality of this operation. In some cases, it may be desirable to allow multiple COD locations to match to the same GADM area, for example Berlin at aggregation level 2, given COD has information on Berliner boroughs but GADM does not have a layer more specific than the entire city, so the best that could be done is to match all the entries about the boroughs to the same singular vertex that would represent Berlin in full. At this time, however, this is unsupported and left as future work. Instead, C19EpidemiologyGraphs currently only allows connections between GADM and COD to happen at a one-to-one cardinality.

The process to join the data sources happens separately for each aggregation level in C19EpidemiologyGraphs. The first step is to search for and concatenate together the most suitable GADM layers for each country, in order to maximize the number of possible matches. This is important in part because while COD only has levels 0, 1 and 2[4], GADM may have more or fewer layers for certain locations, and a suitable GADM layer for a given country or subdivision at a given level might not be the ideal one everywhere else at that level. For example, Belgium has 4 layers in GADM, mapping the country, regions, provinces, and municipalities, but COD only has 3 levels, as they do not use the regional division. As such, when joining level 1, it'd be ideal to use GADM layer 2 for the country of Belgium specifically, while most other countries would stick with GADM layer 1.

When processing level 0, only layer 0 was required for the entire world. However, levels 1 and 2 demanded manual searches to find the suitable layers for each country, if any. After this search, each obtained layer is combined into a single GADM table. For example, at level 2 this table contains layer 2 for most countries, but Chile and Peru required a more specific layer 3, the Philippines required a less specific layer 1, and a few countries, like Canada and Spain, did not have a suitable GADM layer at all. After that, it's possible to join the recently-combined GADM table with COD at the current level.

To create the mapping between COD and these combined layers, there are two sequential steps: name joining and geographic joining. Name joining is performed by finding a unique match across both data sources based on the identifiers of all administrative parent divisions for a given location and the location identifier itself. Countries are checked based on their 3-letters country code, specifically their ISO 3166-1 alpha-3 code, and sub-divisions are checked based on their name. Therefore, to join two countries, all that needs to be done is to check if they have the same country code, but to match two cities, they need to have the same country code, state name (or another intermediate administrative subdivision name), and city name.

After the name join, the rows from both COD and the assembled GADM table

[4]Actually, there are a few entries with aggregation level 3, but only 32 out of a total of 22 963 locations, so they were ignored.

that were successfully matched with a one-to-one cardinality are considered done and do not participate in the geographic join, so that the cardinality constraint is not accidentally broken due to a mix of name joining and geographic joining. The rows that failed to find a one-to-one match remain in their tables and progress to the geographic join step, which checks whether the point defined by the geographical coordinates provided for each location in COD is solitarily contained within the boundaries of a shape in the GADM table. The result of this join must also be one-to-one, discarding matches that aren't. Rows in COD that fail to find a suitable match by both name joining and geographic joining are unable to become vertices in the graphs, and rows that reference them should be discarded.

However, a lot of name joins fail because either one of the names lacks a word (such as "Buenos Aires" in GADM and "Buenos Aires Province" in COD), or because one lacks accentuation while the other has it, or because one is written using the location's language and the other using an English translation. After both joins are performed, instead of discarding the remaining COD rows right away, a manual assessment is first conducted on all unmatched locations to determine whether any of their GADM names could have been changed in order to enable a correct name join. If so, then these changes are applied and both joins are repeated, but this time on a GADM table with said renaming. This process repeats until most viable name changes are applied. The final output is the concatenation of both the name join and geographical join steps. Each one of these one-to-one matches became an entry in C19EpidemiologyGraphs' vertices table.

Of the 22 963 unique reporting locations in COD, 18 690 (81.39%) were successfully matched before the manual name changes. After that transformation, 341 new matches are added, resulting in a final match count of 19 031 (82.87%). A per-level summary can be found in Table 1, and per-country summaries for levels 1 and 2 can be found on this project's GitHub page, due to space constraints.

Table 1. Number of matches grouped by aggregation level

Aggregation Level	Total	Matched	Percentage
0	246	238	96.75%
1	1432	1371	95.74%
2	21 253	17 422	81.97%

3.2. Calculating graph edges from coordinate polygons

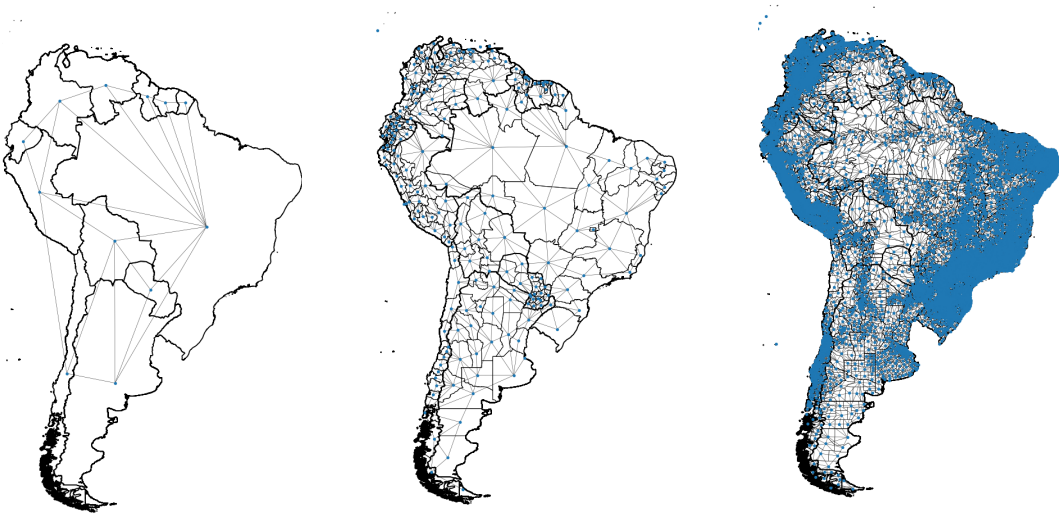
GADM is a database of geographical shapes for administrative areas, made available at various subdivision layers. These shapes are represented as polygons with points written using spatial coordinates. These polygons are used here to create three big graphs, one for each aggregation level in C19EpidemiologyGraphs. For each level in the database, the exact same GADM tables with combined layers that were used to create the connection between the data sources in Section 3.1 are used again here to create the graphs.

To calculate the edges for a given aggregation level, the corresponding GADM table is read into a GeoPandas dataframe, and its internal Sort-Tile-Recursive R-tree data structure is then traversed to check, for each pair of locations in that table, if their polygons

intersect with each other or not. If they do, then they share a border with each other and an undirected edge connecting those two locations is created. Otherwise, they don't share a border and therefore they also don't get an edge. Each undirected edge (v, w) that is found, if inserted into the database, is going to result in two rows: one for (v, w) and another for (w, v), as the edges table stores only directed edges. Also, self loops are not identified by the intersection method and they are absent from the table.

This intersection query is done on all the rows in the GADM table, so all valid edges for that table are discovered. However, only those whose both ends are vertices that were successfully matched with a COD location in Section 3.1 are allowed to be inserted into the table. In other words, in order for an edge to be inserted into the edges table, both its nodes must already be present in the vertices table.

To illustrate what is obtained, Figure 2 shows subgraphs representative of South America, each layered over a map using the original GADM shapes. Do note that the vertices shown in the figure are not all included in the final database, as only those with which COD locations were able to be matched are added. Also, the exact positioning of each vertex might be different when compared to what is present in C19EpidemiologyGraphs, as the database uses the coordinates provided by COD, whereas in the figure, the coordinates are calculated as representative points of the geometrical shape.



(a) Level 0 (b) Level 1 (c) Level 2

Figure 2. Subgraphs representing South America

The number of vertices and directed edges contained within each aggregation level after this filter are shown in Table 2. Said table also shows aggregated metrics related to vertex degrees for the calculated graph, namely their average ($\mathbf{d_{avg}}$), minimum ($\mathbf{d_{min}}$), maximum ($\mathbf{d_{max}}$) and three quartiles ($\mathbf{d_{Q1}}$, $\mathbf{d_{Q2}}$ and $\mathbf{d_{Q3}}$).

3.3. Processing the epidemiological data

The epidemiological source used in this project is COD's epidemiology table, whose rows may contain numbers for confirmed cases, deaths, recoveries and tests performed, both new and cumulative, for a given location and date. C19EpidemiologyGraphs uses that table as a base for its own, replicating all of its columns and only filtering out rows if they

Table 2. Number of vertices and edges grouped by aggregation level

Aggregation Level	#Vertices	#Edges	d_{avg}	d_{min}	d_{Q1}	d_{Q2}	d_{Q3}	d_{max}
0	238	606	2.546	0	0	2	4	13
1	1371	6142	4.480	0	3	4	6	13
2	17 422	92 996	5.338	0	4	5	7	23

don't have a date, a location key or if the location key provided is not in the vertices table (i.e. rows about COD locations that failed to uniquely match with GADM). Beyond that, in order to store a date in SQLite as something other than a text or a blob, the date present in COD's table had to be converted into an integer that tells the number of days that have elapsed since the Unix epoch (January 1st, 1970).

After the filtering is complete, it's now possible to query the epidemiology table to see that, for example, the Brazilian state of Minas Gerais registered 69 844 positive COVID-19 diagnoses and performed 122 497 COVID-19 tests throughout the 10 days from March 5th to March 14th, 2021. On the same period, the Brazilian state of Rio de Janeiro saw 17 026 positive diagnoses out of a total of 59 059 tests. While it was already possible to get this information for both locations by directly processing COD's epidemiology, that would result in two isolated pieces of data, one for each state. However, because the edges table in C19EpidemiologyGraphs uses the same location keys to label vertices as the ones used in the epidemiology table, it's possible to check that Minas Gerais and Rio de Janeiro share a border, as there are edges connecting the two states. This allows users to create spatial connections between the epidemiological pieces of information obtained regarding each location, and perhaps most importantly, to automate this process to create feature graph datasets spanning any desired subset of locations.

C19EpidemiologyGraphs' epidemiology table has 11 543 110 rows. Considering there are 19 031 vertices, this means there is an average of 606.54 epidemiological entries per vertex. Unfortunately, this number is also including rows with mostly null values. Several locations don't tend to report all the data the epidemiology table is capable of holding, and several others may report some information inconsistently, or even just once every several days. In table 3, it's possible to see that the columns that deal with new diagnoses and deaths are very often present, with only 0.424% of rows not having a number for new confirmed cases. Meanwhile, 73.137% of rows don't have information on how many tests were performed on the given day.

There are several options to tackle this issue, one of which is to perform limited interpolation inside sliding time windows. For example, if a user is already using sliding windows of 30 days and needs data on the number of tests, an option could be to interpolate the new_tested column, but keep track of what values are interpolated and what values are real, so that this user is able to impose, for example, a restriction that in a window of 30 days, there can be a maximum of 10 interpolated values. Although hardly ideal, if done well it could lead to window aggregations of better quality. In fact, there is an example notebook in this project's repository that does exactly that: first it interpolates the new_tested column, then it creates a sliding time window and discards any window that exceeds the set threshold for interpolated values or absent days.

Table 3. Nulls in epidemiology table

Column	#NULLs	%
new_confirmed	48 959	0.424%
new_deceased	468 482	4.059%
new_recovered	7 991 177	69.229%
new_tested	8 442 259	73.137%
cumulative_confirmed	197 704	1.713%
cumulative_deceased	660 454	5.722%
cumulative_recovered	7 981 395	69.144%
cumulative_tested	8 622 640	74.699%

Lastly, given that vertices are uniquely identified by COD's location keys, incorporating other tables from COD into this SQLite is almost immediate, as they also use the same keys. For instance, if someone wants to include mobility data, all that needs to be done is to download the .csv, create a new table with a foreign key from the vertices table, and insert the data.

4. Conclusion

This project's main objective was to provide an easy-to-use starting point to build custom graph datasets using COVID-19 epidemiological data, so that C19EpidemiologyGraphs could be queried to fit whatever the end-user's goal is. As such, flexibility for user-defined processing steps was deemed a paramount priority. On that end, we consider this goal to have been achieved successfully.

The epidemiology table fitting into memory allows for a lot of flexibility, as it's possible to load the table, process it using any tabular data manipulation tool and then write it to the database again. All of this culminates in C19EpidemiologyGraphs potentially being a viable pathway towards the creation of epidemiological graph datasets for a variety of purposes, including, but not limited to, time-series analyses and spatial scans of anomalous clusters.

SQLite's ROWID system also allows users to easily switch from textual location keys to unique numerical identifiers, which can then be used to more easily load the assembled graphs into a numerical library, such as SciPy or Torch Geometric, for processing. An example use case is transforming a graph into a SciPy sparse matrix, which can then be used to compute the graph's connected components.

As future work, it could be useful to include some tables from COD other than epidemiology in the database, especially given how straightforward it would be to do so, now that location keys have already been assigned to GADM vertices. Likewise, it could also be useful to perform a similar procedure to incorporate diseases other than COVID-19 into another database.

Usage of Artificial Intelligence

No generative artificial intelligence was used anywhere in this project, neither in the writing of this article nor in the source code.

Acknowledgments

This work was funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, Brazil, grant #307151/2022-0.

References

Deng, S., Wang, S., Rangwala, H., Wang, L., and Ning, Y. (2020). Cola-gnn: Cross-location attention based graph neural networks for long-term ili prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 245–254, New York, NY, USA. Association for Computing Machinery.

Guidotti, E. (2022). A worldwide epidemiological database for covid-19 at fine-grained spatial resolution. *Scientific Data*, 9(1):112.

Guidotti, E. and Ardia, D. (2020). Covid-19 data hub. *Journal of Open Source Software*, 5(51):2376.

Liu, Z., Wan, G., Prakash, B. A., Lau, M. S., and Jin, W. (2024). A review of graph neural networks in epidemic modeling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6577–6587, New York, NY, USA. Association for Computing Machinery.

Wahltinez, O., Cheung, A., Alcantara, R., Cheung, D., Daswani, M., Erlinger, A., Lee, M., Yawalkar, P., Lê, P., Navarro, O. P., Brenner, M., and Murphy, K. (2022). Covid-19 open-data: a global-scale spatially granular meta-dataset for coronavirus disease. *Scientific Data*.

Xie, F., Zhang, Z., Li, L., Zhou, B., and Tan, Y. (2022). Epignn: Exploring spatial transmission with graph neural network for regional epidemic forecasting.