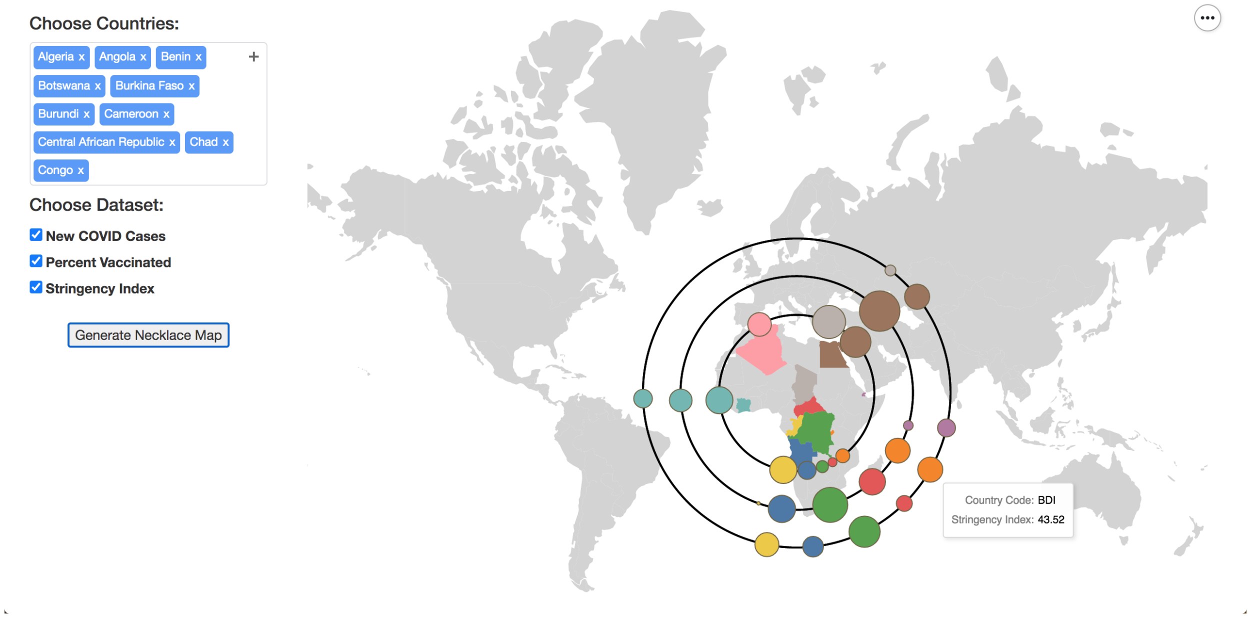
Necklace Maps for COVID-19 Visualization

Mary Abikoye, Minglong Li, Jocelyn Minns


Choose Countries:

Algeria x Angola x Benin x Botswana x Burkina Faso x Burundi x Cameroon x Central African Republic x Chad x Congo x

Choose Dataset:

- [✓] New COVID Cases
- [✓] Percent Vaccinated
- [✓] Stringency Index

Generate Necklace Map

Country Code: BDI
Stringency Index: 43.52

Fig. 1. A view of the necklace map generation tool. Countries and data specified on the left will be plotted as symbols on appropriate necklaces on the right. Hovering over a symbol allows the user to see exact data about that country.

Abstract— Choropleth maps are commonly used for visualizing epidemiological data, however their weakness lies in the ability give visual weight to large geographic areas regardless of their significant on the data. Necklace maps offers an alternative that shifts the visual weight to the data attributes rather than the geographic area. In this project, we implement the automated necklace map generation algorithm while incorporating an interface to allow users to generate necklaces that correspond to specific countries and data attributes. We extend the original algorithm to allow multiple necklaces to visualize multiple data attributes at once. We looks at the resulting necklaces maps for readability and examine the computational time required with this implementation to further understand the usability of this approach.

Index Terms—Necklace Maps, Proportional Symbol Maps, Epidemiology Visualization

1 Introduction

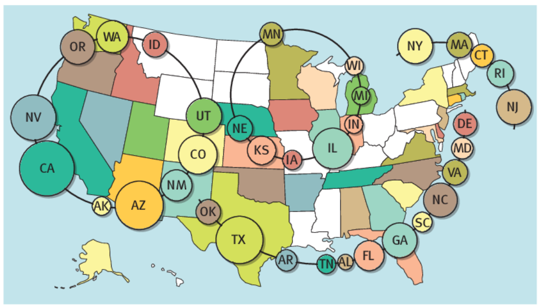


Fig. 2. A Figure from the original necklace map paper [8]

Necklace maps [8], as shown in Figure 2, are a form of visualization that combines elements of proportional symbols maps and boundary labeling. The projection maps each region of the input to a contiguous interval on the necklace in such a way that the interval captures the global location of the region with respect to the necklace.

The advantage of necklace maps over the common choropleth maps is that they give more visual weight to the quantitative variable, rather than just the geographic area. This can be preferred when we are dealing with epidemiological data since large geographic areas may have a space population, but that large area holds more visual weight than a densely populated city. When users see these choropleth maps, they can easily misunderstand the data to assume the situation is better or worse than the quantitative variable indicates.

Since the rise of the COVID-19 pandemic in early 2020, data about various factors in the spread and response to the disease have become commonplace in our daily lives. However, many visualizations of this data continue to favour choropleth maps. For this project we implemented necklace maps to visualize three major factors of the COVID-19 pandemic: confirmed COVID-19 cases, vaccination rates and a stringency index.

- *Mary Abikoye. E-mail: temiabik@student.ubc.ca.*
- *Minglong Li. E-mail: limnglng@cs.ubc.ca.*
- *Jocelyn Minns. E-mail: jminns@cs.ubc.ca.*

2 Related Work

Choropleth maps are a thematic map that is well suited to show quantitative attributes of geographic data [6], however there can be significant drawbacks to relying on this method. When the quantitative attribute is focused on the population, these maps can overemphasize large geographic areas regardless of their sparse population due to their strong visual weight. Different thematic maps have been proposed over the years and while they may each have their own advantages, no clear alternative has been identified [1]. Additionally, choropleth maps can be ill suited for the task of observing changes over time in the data. In a study on animated choropleth maps, the phenomenon of change blindness hindered the user from perceiving changes in the data [3].

Epidemiological data is commonly visualized using choropleth maps, though careful consideration must be given to ensure these maps can be useful to epidemiologists. Variations in classifying the data must be considered to accurately inform those relying on these visualizations [2]. The recent COVID-19 pandemic has seen a surge in visualization approaches [9] including a heavy reliance on choropleth maps, however the limitations of these maps must be considered when creating visualizations to avoid potential misinformation [5].

Necklace maps [8] are a proposed alternative to the choropleth maps. This method combines proportional symbol maps and boundary labeling to create a one-dimensional curve containing scaled symbols corresponding to a quantitative attribute for a particular region. This can help visualize data that is not proportional to region sizes.

3 Data and Task Abstraction

To evaluate our implementation, we will focus on specific datasets and tasks while producing our necklace maps.

3.1 Datasets

Our World in Data provides publicly available datasets on the COVID-19 pandemic [7]. Specifically, we want to look at three attributes: Confirmed COVID-19 cases, reported vaccination rates and the stringency index.

The confirmed COVID-19 case data gives a daily update of the confirmed case count per million people by country. This data is updated daily for many countries, but reported cases can vary depending on how the area chooses to report their cases on weekends or holidays. To mitigate this reporting inconsistency, we will use the smooth data over the 7-day period. Vaccination rates provide the reported percentage of people vacationed by country. We will focus on the data of people both partially and fully vaccinated. The stringency index looks at the response policies regarding the COVID-19 pandemic. The Oxford Coronavirus Government Response Tracker (OxCGRT) [4] project calculates this stringency index of countries using 20 indicators divided into 5 categories: Containment and closure policies, Economic policies, Health system policies, Vaccination policies, and Miscellaneous policies. The final stringency index is a number between 0 and 100 where 100 is the strictest.

3.2 Tasks

The main tasks we hope to accomplish will involve visualizing the datasets listed above. Some tasks we would like to achieve with our implementation of the necklace map algorithm are:

- Discovering trends by comparing symbols on different necklaces. e.g. Is there a noticeable pattern in the case number necklace and the stringency index necklace?
- Look up a specific country's data. e.g. What are the current COVID-19 case numbers in Canada?
- Compare different geographic areas: e.g. How does the vaccination rate necklace of Europe compare to the vaccination rate necklace of North America?

4 Scope

In this section we define the relationship between our work and the original necklace map paper [8]. In addition to generating a visually pleasant necklace of glyphs representing quantitative data, our work extends the capabilities of the necklace map mostly in three aspects: interactive environment, dynamic necklace generation and multiple-necklace display.

4.1 Interactive Environment

The original necklace map is a static visualization, however, we would like to present an interactive map to let the user to freely explore any set of countries with available data. Also, since for each country we have three sets of quantitative data (case numbers, vaccination rate and policy stringency index), the user should also be able to select any or all three datasets to visualize.

4.2 Dynamic Necklace Generation

Speckmann and Verbeek [8] generated the necklace manually first and then placed the glyphs onto calculated spots. However, in our case, the radius and center of the necklaces will be determined automatically based on the selected countriesin.

4.3 Multiple-necklace Display

In order to visualize a maximum of three quantities per country at the same time, we instantiate one necklace per quantity. Therefore, if the user selects to visualize all three datasets, there will be three necklaces at the same time, as opposed to one fixed necklace [8].

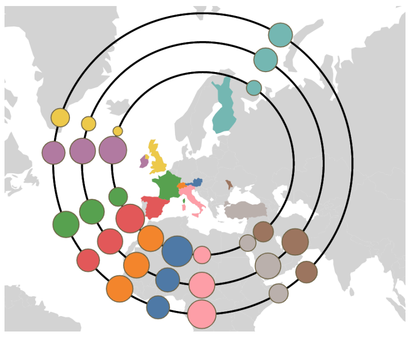


Fig. 3. Nested necklaces used to show multiple data attributes as symbols on multiple necklaces. Each necklaces represents a different attributes.

We considered several ways to visualize multiple datasets with the necklace map algorithm. The simplest way is to split the screen into different regions where each region contains an entire map with a single necklace in it. Each region would essentially be responsible for visualizing one of the datasets. There are two main drawbacks to this method. First, the user might not always want to visualize the same number of datasets, which means the division of the screen is dynamic, thus poses more cognitive load on the user. More importantly, since each region contains an entire map of the world, smaller countries become harder to identify. The geographical association would not be preserved since it relies on the colour coding between countries and markers.

Another approach is to use some type of glyph to represent all the datasets of interest of the same country and place them on one necklace. The advantage is that it saves a lot of space on the screen so it allows potentially larger necklaces. However, the choice of glyph becomes rather difficult, since the datasets to be visualized are not always directly comparable to each other. For example, the vaccination percentage and COVID-19 case number per million people of a certain country have different units. Therefore, it does not make sense to combine them in a bar chart or pie chart. Other choices of glyphs such as mini heatmap or some sort of texture glyphs would compromise the quantitative interpretability of necklace map.

The approach that we use, as mentioned before and shown in Figure 3, is to use concentric necklaces to represent different datasets and show all of the necklaces on the same map. The markers are still circles so the quantities are still encoded with the area of the circles. This encoding means that the quantitative data can be interpreted rather easily by the user. In addition, the nature of the algorithm makes markers of the same country on different necklaces appear roughly on the same radial line. This fact combined with colour coding allows the user to associate the countries across different necklaces.

5 Implementation

In this section we discuss our implementation decisions and details, by breaking down our approach to the problem into the following aspects.

5.1 Necklace Map Generation

5.1.1 Language and Packages

In order to implement the Necklace map algorithm (backend of the project), it is vital that we select the appropriate programming language and packages. After experimenting with some of the recommended options, we decided to go with Python due to our familiarity with it. To build an interactive visualization, we are using the recommended Altair package. In addition, Geopandas is utilized for its easy access to geographical information and Pandas to read and store the COVID-19 dataset.

5.1.2 Retrieving Data

The data for both the geographic polygon and COVID-19 datasets will be loaded into the Python script. Geopandas and Pandas provides a simple interfaces to retrieve the data and store the in their respected data frames. Both datasets contain an attribute of the county's international ISO 3166 standard code which we use to associate the COVID-19 data with each geographic polygon. The COVID-19 dataset is retrieved from the Our World in Data website each time the Python script is executed to ensure the necklaces contains the most recent data.

5.1.3 Necklace Generation

Automatic generation of necklaces based on user selection is something that was not included in the necklace paper. We devise an algorithm for this purpose as follow:

1. Find the convex hull encompassing all countries and its geometric center. This step can be accomplished quite straightforwardly with the geographical data provided by Geopandas.

2. We choose the geometric center of the convex hull as the center of the necklace. The radius of the innermost necklace can be determined by finding the largest distance between vertices of the convex hull and the center.

3. In the case where the user selects to visualize multiple variables, the size of the markers on the inner necklace are calculated first and the maximum radius is used as the gap between inner necklace and the outer necklace.

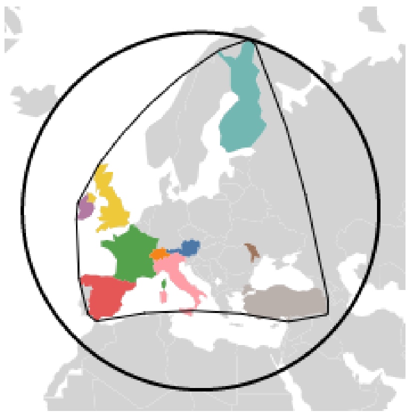


Fig. 4. A visualization of a convex hull and its encompassing necklace.

An example showing the convex hull and the innermost necklace can be found here in Figure 4. Note that this is different from the smallest circle surrounding the convex hull. The center is shifted towards area with more countries to be more visually pleasing.

5.1.4 Marker Size

To calculate the size of the markers, we use the same algorithm described by Speckmann and Verbeek [8]. For the purpose of explanation, we define a dataset $Z = \{z_1, z_2, \ldots z_n\}$ where z_i is the normalized numerical value of the variable that the user wants to visualize on the necklace for country i.

We begin by calculating the feasible interval, I_i, for each country i. A feasible interval is the maximum arc a marker is allowed to be placed within along the necklace. The original paper provided three different approaches to calculating this interval. Our implementation uses the Centroid Intervals approach which calculates the center of the feasible interval using a line from the the center of the necklace and through the centroid of the county and intersecting with the necklace perimeter. The interval is a fixed length depending on the number of markers that will be placed on the necklace. Few markers will allow larger intervals per marker, while increasing the amount of marker will decrease the interval size. Note that feasible intervals for different markers can overlap. For detailed description of how it is calculated, we refer readers to Speckmann and Verbeek [8].

Then, to make the area of the marker of country i proportional to z_i, we calculate the radius of the marker as

$$z'_i = asin(\frac{\rho\sqrt{z_i}}{r})$$

where r is the radius of the necklace and ρ is a coefficient that is the same across all markers [8].

The last step of optimizing ρ is where our implementation differs from the original implementation. The goal of this step is to adjust ρ such that all markers do not overlap and take up as much space as possible on the necklace while their center remain within their feasible intervals. The original implementation approximates z'_i using a linear relationship, $z'_i = \rho z_i$, and obtain ρ by solving a linear programming optimization problem.

However, in our case, the total number of selected countries is always a small number. Therefore, we can afford to use the more accurate representation $z'_i = asin(\frac{\rho\sqrt{z_i}}{r})$. In our implementation, we first calculate the upper bound ρ_{max} by setting the markers to take up the entire necklace. Then, we perform a binary search on ρ with minimum being 0 and maximum being ρ_{max}. The search ends when all markers remain in their intervals and at least one marker lies on the far edge of their feasible interval. At this point we can determine this as our ρ_{max} since any larger ρ would push that marker outside of their interval.

If the data is not available for a specified country, then no marker will be included on the necklace for the country. When more than one necklace is generated, symbols on outer necklaces will be sized assuming they would be placed on the inner most necklace. This is to prevent outer necklaces from holding more visual weight that inner necklaces, but does result in outer necklaces having larger spacing between symbols.

5.1.5 Marker Placement

After the sizes of the markers are determined, we use the same approach in the original necklace paper [8] to calculate the positions of the markers. The idea is to assume that there are two forces acting on each marker: a repelling force F_{rep} and a centering force F_{mid}. The repelling force comes from adjacent markers and the centering force tries to keep the marker to the center of its feasible interval. The original implementation sets the total force of all markers to be 0 and solves this system of equations.

However, in our implementation, the problem is relaxed to be an optimization problem where we minimize the sum of absolute value of all the net forces, with the feasible interval being the constraints for the positions. This relaxation allows for more feasibility even when dealing with regions with extreme shapes and arbitrary arrangement of those regions.

5.2 User Interaction

The user interface is implemented with HTML, JavaScript and PHP to allow users to generate specific maps. Unfortunately the Altair toolkit does not suppose direct user interaction for geoshape charts, thus we were not able to allow users to select countries on the map itself. Instead users may select countries from the search bar provided and choose data attributes to visualize from the checkboxes. We limit the number of countries a user can request to 10 since more would degrade the colour encoding on the necklace. Once the user enters their designed data, JavaScript must pass this from the client-side to the server-side for the necklaces to be computed and generated. To do this, we use a PHP script to act as a bridge between the JavaScript and the Python programs. The PHP script executes the necklace map generation script and replies to the client-side when the process is complete. The JavaScript can then retrieve the resulting chart as JSON data from the server to display to the user, as shown in Figure 5. This process is unfortunately time consuming and takes several seconds to complete.

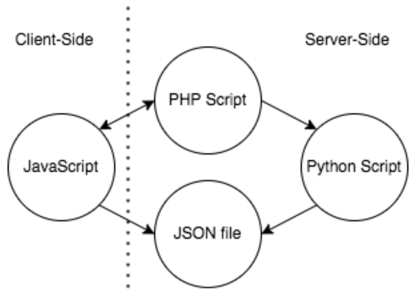
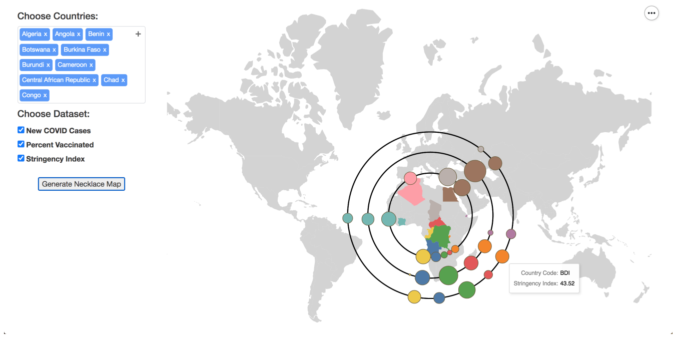


Fig. 5. A visualization of the flow of information between the client-side and the server-side while generating a necklace map.

6 Results


Choose Countries:

Algeria x · Angola x · Benin x · Botswana x · Burkina Faso x · Burundi x · Cameroon x · Central African Republic x · Chad x · Congo x

Choose Dataset:

- ✓ New COVID Cases
- ✓ Percent Vaccinated
- ✓ Stringency Index

Country Code: BDI
Stringency Index: 43.52

Fig. 6. The interface of the resulting program.

Figure 6 shows the interface of the resulting program of this project. On the top left is a dropdown search bar where the user can type to search or scroll and select their countries of interest. Below the search bar are checkboxes for the datasets. On the right is the main view which shows an entire map of the world projected using the Mercator method.

Below we describe the ideal final program and how the user would interact with the tool. We aim to have the users explore the dataset with the following steps:

1. Upon starting the software, a map of all countries in the world appears. The user can hover over the countries to view all available data for that country.

2. The user would then use the search bar to select countries and datasets they want to visualize.

3. After selection is finished, the user can click the "Generate Necklace Map" button to view the necklace(s). For each dataset they selected, a necklace visualizing the data of interest would appear in concentric circles.

4. All selected countries are colour coded corresponding to its symbol colour on the necklace. And all unselected countries would be grey.

We benchmarked our algorithm on the linux virtual machine with the following specifications: Intel Xeon CPU E5620 @ 2.40GHz with a memory of 16G. Table 1 shows the runtimes of different configurations:

Number of Necklaces	Number of Countries	Average Runtime (s) Over 10 Runs
1	2	6.7862
1	5	6.7461
1	10	6.7666
2	10	8.0949
3	10	8.8818

Table 1. Benchmark of our algorithm

Since the maximum number of countries that can be visualized at the same time is limited to 10, our benchmark only goes up to 10 markers. The resulting map can be generated within the matter of seconds, which means the user can explore the datasets at a reasonable speed.

7 Milestones

See Appendix A.

8 Discussion and Future Work

Previously no user-interface allowed for dynamic necklace map generation which limited many from using this type of proportional symbol map. This implementation may allow other users to explore necklace maps to see the benefit of alternative types of geographic visualization. By allowing the extension of multiple necklaces to visualize different attributes, we also extend the tasks that can be achieved using this algorithmic approach.

However, there are limitations to our implementation. Our implementation suffers from a slow user-interaction design. Since it will take several seconds to connect to the backend code, load the full datasets and produce a necklace, users can get frustrated with the system. Much of this is from the design of the communication between the frontend and backend of the system.

Another limitations is that necklaces which contain countries that are geographically far apart will produce visual challenges. If a necklace surrounds distant countries then it will become too large and overflows on the left or right of the map. This will cause the necklace to wraparound to the other side of the map creating a necklace that is unusable to the user due to its disjointed display.

Lastly since we choose to use the Altair toolkit, there is little user interaction allowed with the produced necklace map. The intuitive interaction would allow users to directly select which countries on the map that they would like visualized on the necklace, however we limit users to the less desirable design of selecting countries through a search bar. This is due to the static nature of Altair visualizations.

8.1 Lessons Learned

Thought the development of our necklace map implementation, we have learned the importance of thoroughly investigating toolkits early in development. Had we realized earlier the limitation of Altair, we may have opted for a more interactive visualizations rather than limiting how users were able to interact. We would need to choose a toolkit that better balances the computational requirements and the interactivity.

8.2 Future work

To improve our necklace map implementation, there are several areas we would want to focus on going forward. A key feature of the original algorithm that we were not able to implement ourselves is the use of irregular necklace shapes. If we introduced this feature, then we could better deal with countries that are far apart that may produce overly large necklaces by opting for a curve or shape that more closely follows the geographic area.

Another area would would like to improve is the ability to visualize many countries at once. If we intruded a clustering algorithm then we could spilt symbols between multiple necklaces to keep them close to their geographic area while not overloading a single necklace.

We would like to optimize the time to generate the requested necklace map. One approach would be to use a Python server running that we could be connected to an I/O socket rather than depending on a PHP script to execute the Python code each time. This would decrease the time since the script would not need to load the datasets from Our World in Data each time a necklace is produced. More importantly though we would want to focus on optimizing the algorithms used in the necklace map computation. Due to time limitations of this project, we opted for simplified implementations of the optimization algorithms which we cannot guarantee are the most efficient implementations.

9 Conclusion

Necklace maps offer a unique way of visualizing and interpreting epidemiological data. By implementing this type of visualization with the COVID-19 dataset, we can allow users further explore the advantages of different types of proportional symbol maps. We outlined the areas where we observed the original automated necklace map germination algorithms and where we differed to complete the same requirements. With the resulting implementation, we are able to proceed multiple necklaces to visualize multiple attributes. Lastly, we are able to do this within a reasonable computational speed that allows users to interact with the system to produce customized necklace configurations.

Acknowledgments

We would like to thank professor Tamara Munzner for her instruction and guidance throughout the term. We would also like to acknowledge Bettina Speckmann and Kevin Verbeek who authered the paper that inspired this project.

References

[1] L. Besançon, M. Cooper, A. Ynnerman, and F. Vernier. An evaluation of visualization methods for population statistics based on choropleth maps. *arXiv preprint arXiv:2005.00324*, 2020.

[2] C. A. Brewer and L. Pickle. Evaluation of methods for classifying epidemiological data on choropleth maps in series. *Annals of the Association of American Geographers*, 92(4):662–681, 2002.

[3] C. Fish, K. P. Goldsberry, and S. Battersby. Change blindness in animated choropleth maps: An empirical study. *Cartography and Geographic Information Science*, 38(4):350–362, 2011.

[4] T. Hale, N. Angrist, R. Goldszmidt, B. Kira, A. Petherick, T. Phillips, S. Webster, E. Cameron-Blake, L. Hallas, S. Majumdar, et al. A global panel database of pandemic policies (oxford covid-19 government response tracker). *Nature Human Behaviour*, 5(4):529–538, 2021.

[5] C. Juergens. Trustworthy covid-19 mapping: Geo-spatial data literacy aspects of choropleth maps. *KN-journal of cartography and geographic information*, 70(4):155–161, 2020.

[6] T. Munzner. *Visualization Analysis and Design*. CRC Press, Taylor & Francis Group, 2015.

[7] H. Ritchie, E. Mathieu, L. Rodés-Guirao, C. Appel, C. Giattino, E. Ortiz-Ospina, J. Hasell, B. Macdonald, D. Beltekian, and M. Roser. Coronavirus pandemic (covid-19). *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[8] B. Speckmann and K. Verbeek. Necklace maps. *IEEE Trans. Vis. Comput. Graph.*, 16(6):881–889, 2010.

[9] Y. Zhang, Y. Sun, L. Padilla, S. Barua, E. Bertini, and A. G. Parker. Mapping the landscape of covid-19 crisis visualizations. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–23, 2021.

A Milestones

Weeks	Task and Description	Starting Data	Actual Hour Spent
W1 (Oct 17th - Oct 23rd)	• T1: Write Project Proposal (Oct 21 noon) (all members - 4 hours each)	Oct 17	T1: 4 hours each member
W2 (Oct 24th - Oct 30th)	• T1: Literature Research (all members - 4 hours each) – Other map visualization techniques – Capabilities and learning curves of multiple tools • T2: Data Exploration (all members - 1 hours each) – Examine data available. Determine what filtering is required from existing data: Stringency Index Case Number Vaccination Percentage	Oct 24th	T1 & T2: 5 hours each member
W3 (Oct 31th to Nov 6th)	• T1: Programming Ramp-Up (all members - 6 hours each) – Become familiar with chosen programming tools – Explore capabilities and limitations of Altair • T2: Geometric Map Visualization (Jocelyn - 6 hours) – Visualize the world map – Link database with the application – Implement choropleth maps to explore compatibility of the geographic data and COVID-19 datasets	Oct 31th	T1: 6 hours each member T2: Jocelyn - 6 hours
W4 (Nov 7th to Nov 13th)	• T1: Establish Necklace Algorithm Implementation (Minglong - 6 hours) – Determine convex hull for necklace shape – Plot proportional symbols given our dataset	Nov 7th	T1: Minglong - 4 hours
W5 (Nov 14th to Nov 20th)	• T1: **Write Project Updates (Nov 16 3pm)** (all members - 4 hours each) – Discuss feedback – Reevaluate scope of the project • **Peer Reviews (Nov 17 in class)**	Nov 14th	T1: 4 hours each member

Weeks	Task and Description	Starting Data	Actual Hour Spent
W6 (Nov 21st to Nov 27th)	• **Post-update Meetings (Nov 24 & other times that week TBD)** • T1: Modification Based on Comments (all members - 1 hours each) – Make necessary modifications and adjust plans based on reviews from the class and the comments from the meeting • T2: Complete Necklace Algorithm Implementation (all members - 6 hours each) – Hard code pre-defined clusters of countries – Draw necklaces for each cluster and dataset	Nov 21st	T1: 1 hour each member T2: 3 hours each member
W7 (Nov 28th to Dec 4th)	• T1: Advanced Features (all members - 6 hours each) – Design and implement automatic generation of the necklaces – Enable user interaction to select countries and datasets	Nov 28th	T1: 4 hours each member
W8 (Dec 5th to Dec 11th)	• T1: Advanced Features Cont. (all members - 6 hours each) – Finish up previous features – Implement country search bar if time allows	Dec 5th	T1: 10 hour each member
W9 (Dec 12th to Dec 17th)	• Final code base testing and debugging • Pre-recording the final presentation • **Final Presentation (Dec 15 2-6pm)** • **Final Paper (Dec 17 8pm)**	Dec 13th	5 hours each member