FlowMapper.org: A web-based and interactive framework for designing origin-destination flow maps

Caglar Koylu[1]*, Geng Tian[1], Mary Windsor

*Corresponding author: caglar-koylu@uiowa.edu

1 Geographical and Sustainability Sciences, University of Iowa, Jessup Hall, Iowa City, IA 52242

Abstract

FlowMapper.org is an interactive web-based framework for automated production and design of origin-destination flow maps (https://flowmapper.org). FlowMapper has four major contributions. First, FlowMapper allows the user to upload their own flow data to design, produce and share flow maps with customized symbology with a variety of interface elements and options. Second, we introduce alternative flow path symbology by drawing curved flow paths with varying thickness along a flow line, which reduces the visual cluttering and support different flow map readings tasks such as flow magnitudes, direction, and clustering. Third, FlowMapper supports additional layers to contextualize flow patterns with locational characteristics such as net-flow, total flow, or a relevant locational attribute. It supports user interactions to zoom, filter, and obtain details-on-demand functions to support visual information seeking. Finally, the web-based architecture of FlowMapper supports server-side computational capabilities to process and summarize large flow data to reveal natural patterns of flows.

Keywords: Flow mapping, web mapping, visual analytics, interactive cartography, spatial interactions

1. Introduction

Flow maps illustrate movements of tangible and intangible phenomenon in between locations. While there are many types of flow maps, origin-destination (OD) flow maps illustrate directed flows between origin and destination locations while ignoring the actual routes of connections between locations (Slocum et al., Dent, Torguson, & Hodler, 2008; 2009; Tobler, 1987). OD flow maps use several visual variables including flow line thickness and color value to indicate volume of flows, arrows, half-arrows, origin-destination coloring, or line styles to indicate direction between locations.

Flow map comprehension is challenging due to the visual cluttering of flows, even in small data sets. For example, interstate migration within the U.S. between 2015 and 2020 form a network of state-to-state migration flows, which consists of 50 nodes (states), thousands of links and millions of migrants moving through those links between the states. If there are flows between every pair of states in each direction, there would be a total of 50 x 49 = 2450 flow lines. Displaying all flows makes it impossible for map reader to understand flow patterns. Common methods for addressing visual cluttering is selective filtering such as displaying flows that exceed a threshold value, the top 100 flows, or flows to and from a particular location (Tobler, 1987). However, filtering lacks the ability to produce an overview of the flow patterns while clearly identifying patterns on specific subset of flows. Edge bundling is also commonly applied to bundle flow lines to reduce cluttering and improve clarity (Graser et al., 2019; D. Holten & van Wijk, 2009; Phan et al., 2005). While edge bundling substantially reduces cluttering, it makes it difficult for the map reader to follow the connections between origins and destinations, and map readers may wrongly interpret the bundled edges as actual routes of flows (von Landesberger et al., 2016). Computational methods that summarize and aggregate flows prior to making flow maps are also common. Flow-based regionalization (Guo, 2009), location-based clustering (Andrienko et al., 2017; von Landesberger et al., 2016), and flow data smoothing and clustering (Guo & Zhu, 2014; Tao & Thill, 2016; Zhu, Guo, Koylu, & Chen, 2019) are commonly used methods to summarize flows. Clustering and regionalization methods aggregate individual flows into flows between contiguous larger regions or clusters of locations, therefore, reduce the number of regions and flows to be displayed by aggregating places and flows. Flow clustering and density estimation methods also reduce the number of flows by identifying and visualizing representative or significant flows rather than displaying all flows. Flow clustering methods show great promise for summarizing large flow data sets, providing an overview of patterns with multi-scale mapping ability. However, these methods are computationally expensive, and currently, may be provided as asynchronous processing tools on a web-based flow mapping environment.

As compared to straight flow lines, curved flow lines have been shown to enhance readability in graph visualization (Danny Holten, Isenberg, Fekete, & Van Wijk, 2010) and flow maps (Jenny et al., 2018; Koylu, 2014). Following previous research in graph visualization and cartography, Jenny *et al.* (2018) outlined design principles for curving of flows in origin-destination flow maps. These principles include minimizing flow line intersections, obtaining large angles at intersections, drawing symmetric single and gradual curves, avoiding flows to pass through nodes and narrow angles between flows at shared nodes. Jenny *et al. (2017)* utilized force-directed graph drawing algorithm to follow these principles and create OD flow maps that depict a one-way connection (net flow) between two locations. Due to the increased number of flow lines, force-directed algorithm is less effective for satisfying the design principles for directed two-way flow visualizations in which there are two flows between a pair of locations such as a flow from location A to B and from B to A. In addition, some of these design principles may be less effective depending on the type of map reading task. For example, although symmetric curves may enhance flow map readibility for identifying flow connections, magnitudes, and clustering, asymmetric curvature is shown to enhance the perception of flow directions (Koylu & Guo, 2017; Ware, Kelley, & Pilar, 2014). Both studies suggest that the best use of flow visualization symbolization is dependent on the data and scale because they affect how user scans the map and which features are visually salient. A flow map eye tracking study (Dong, Wang, Chen, & Meng, 2018) found that users could more accurately interpret curved flows over straight flows. Dong *et al.* (2018) also found that flows the use of a color gradient to represent flow volume were more effective for user's interpretation of flows that differed by line thickness.

There have been recent advancements in web-based automated flow mapping applications. Stephen & Jenny (2017) designed a flow map application that allows users to provide an overview for flow patterns with a specific application in U.S. domestic migration. Stephen and Jenny's (2017) US migration flow mapper provides not only an overview of state-to-state migration flows but also details-on-demand functionality that allow users to select and visualize flows to and from counties in a selected state. The US migration flow mapper allows selecting a state and visualizing county-to-county flows within that state and flows between the counties within the state and other states using a circular layout that projects the states as nodes around a selected state. Nost et al. (2017) visualize hazardous waste flow data in their HazMatMapper to explore potential environmental justice issues. With HazMatMapper, users are also able to obtain an overview first, and then filter the dataset in greater detail through controls at the state level for a regional view. HazMatMapper includes an information panel that provides additional statistics to the user. Flowmap.blue is a flow map application that allows the user to cluster flows and animate flows if temporal information is available (Boyandin, 2021). The flowmap.blue application allows rendering of large number of flow lines using WebGL (Parisi, 2012) and allows location clustering to aggregate flows between clusters that are formed by nearby units. Flowmap.blue provides a dark grey base map and straight flow lines with half-arrows at the end to depict flow direction. Users can select certain nodes to highlight the flows connected to that node. Aside from flow mapping applications, there are also other novel web-based applications that contribute to interactive cartography and web mapping. Lan, Delmelle, and Delmelle (2021) feature the Neighborhood Dynamic System, which is developed around a time-based data set of neighborhoods, that includes a temporal sliding bar, bar charts, a Self-Organizing Map, and map comparisons, all of which help users examine the spatial-temporal dataset in more detail. Lan and Longley (2021) developed a web application that focuses on aggregating and summarizing geodemographic information, and address scale-dependent issues in the analysis and visualization of geodemographic structure of functional regions.

Despite the recent advances in design principles, usability evaluation and applications of flow maps, automated techniques for web-based flow mapping are limited and application specific. This is because of several challenges for producing generic web-based flow map applications. First, spatial interaction or movement data are often very large, which requires computational summarization methods to process flow data and reveal natural flow patterns. Such transformation needs to be done prior to creating flow mapping, and on-the-fly transformation of flow data is computationally expensive especially on web-based environments. Second, flow maps convey users different types of information such as flow volumes (or magnitude), directions, clustering and spatial focusing of flows. These different types of information often require different symbology choices to best communicate the type of pattern to the map reader. It is difficult to design complex interfaces that can enable and guide users to adjust symbology and design for visualizing flow patterns for different tasks. Third, designing a user-friendly and accessible interface for the general public is challenging due to the interface complexity and various symbology choices for flow map design.

In this article, we introduce a web-based framework, https://flowmapper.org, to produce and share two-way directed OD flow maps with various design choices. FlowMapper has four major contributions. First, FlowMapper allows the user to upload their own flow data to design, produce and share flow maps with customized symbology with a variety of interface elements and options. Second, we introduce alternative flow path symbology by drawing curved flow paths with varying thickness along a flow line, which reduces the visual cluttering and overlapping at the origin and destination locations. Alternative symbology choices allow the user to support different flow map reading tasks such as comparing flow magnitudes and directions and identifying flow and location clusters that are strongly connected with each other. Third, FlowMapper supports additional layers to contextualize flow patterns with locational characteristics such as net-flow, total flow, or a relevant locational attribute. It supports user interactions to zoom, filter, and obtain details-on-demand functions to support visual information seeking. Finally, the web-based architecture of FlowMapper supports server-side computational capabilities to process and summarize large flow data to reveal natural patterns of flows.

2. Map design

To support a variety of flow map reading tasks, Flow Mapper consists of a flow map with alternative flow line designs, a point symbol map, a choropleth map, and a reference base map.

2.1. Flow map

FlowMapper provides users a wide range of choices for determining flow line symbology which consists of the specific algorithm and design of flow lines, flow line width and coloring.

2.1.1. Flow line style

Force-directed layout algorithm used by Jenny *et al.* (2017) is less effective for reducing visual cluttering in two-way directed flow maps in which there are two flow connections between a pair of locations. This is because the graph drawing algorithm cannot effectively support the design principles of crossing angles and minimizing of intersections when there are many more flow lines especially two flow lines between the same pair of nodes. Moreover, asymmetric curvature has been found to be effective in direction reading tasks in two-way flow maps. We introduce algorithms for drawing curved or straight flow paths that allows the variation of the line width along a flow line. Unlike monotone line thickness along the flow line, FlowMapper's algorithms reduce the overlapping of flow lines at the origin and destination locations. FlowMapper currently provides four flow line designs: Curve with half-arrow, tapered curve, teardrop curve, and straight-line with half-arrow (Figure 1). All the curves are asymmetric to enhance readability of directions in two-way flow maps (Koylu & Guo, 2017). The algorithms and directions for producing the cubic Bezier flow symbols are explained in detail in Appendices section 1. Other alternative symbols including origin-destination coloring, partial lines and symmetric curves and curves with varying asymmetry will be included in feature revisions of FlowMapper.

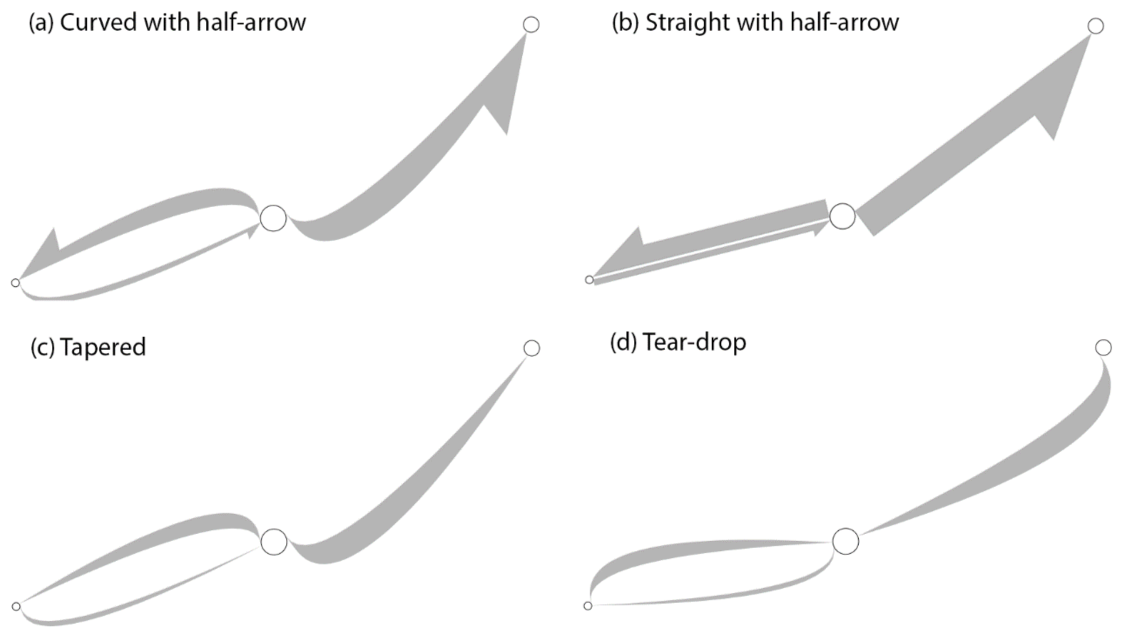


Figure 1: Flow line styles (a) Curve with half-arrow (b) Straight with half-arrow (c) Tapered (d) Teardrop

2.1.2. Flow line (path) thickness

To determine flow line thickness, users can choose a proportional scale in which flow thickness is proportional to flow magnitude or volume. Perceptual issues related to proportional symbol mapping are also inherent in proportional flow thickness - it is often hard to perceive differences between flows on a continuous scale. The effect of different line length also make longer flows to be visually more salient, which is a similar problem to the perception of large areas in choropleth maps. Also, there are only a few large flows and many small flows in most flow data sets, which makes it challenging for comparing flow volumes. Similar to graduated (or range-graded) point symbols, FlowMapper provides a graduated flow symbol for classifying flow volumes using commonly used classification methods such as natural breaks, equal interval, quantile, and manual classification.

2.1.3. Flow line color

We use flow filling color under the curve as a double visual variable (in addition to flow line width) to visualize flow magnitude. Users can select a classification method and choose a color scheme from colorbrewer.org. The user can also choose an unclassed (or continuous) color scheme using min-max scaling of values to two colors for minimum and maximum flow magnitudes. The user can also choose to use a flow stroke. Flow stroke enables easier perception of flow lines on areas where multiple flow lines crossing over. In addition to choosing a color gradient, the user can employ a single color for depicting flows with different magnitude. Figure 2.a illustrates a proportional flow symbol legend with a single color (black), while Figure 2.b illustrates a flow symbol legend with a continuous color scheme from white to blue. The largest and smallest flow lines are designed by the user, and an average flow line symbol is shown for the user as a reference to an average flow magnitude for the proportional flow symbols. Figure 2.c illustrates a classified flow magnitude using a quantile classification with 5 classes. Classification is applied to both flow colors and widths.

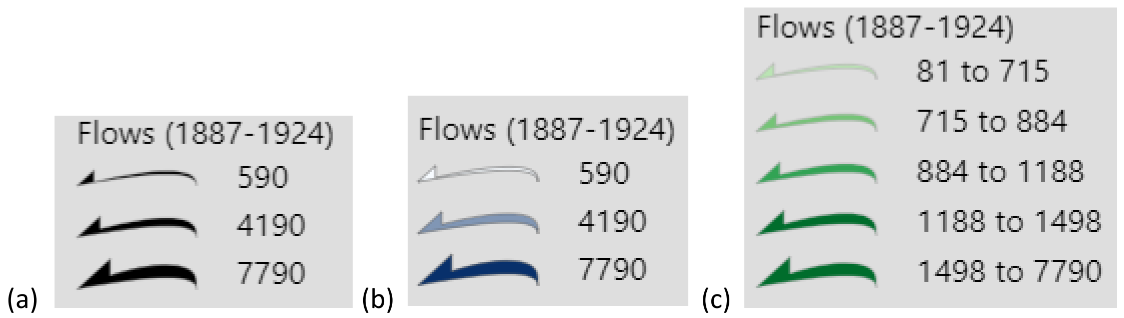


Figure 2: Scaling and legend alternatives for flow symbols (a) a single color (black) proportional flow line symbol, which only scales line thickness to be proportional to flow volume. (b) Flow symbol with both the color and thickness are scaled proportional to flow volume (b) a classified flow legend which uses five classes assigned by a quantile classification.

2.2. Node symbol map

The choice of whether to use node symbols depend on the data set and task. For example, airline flow data represent passenger flows from and to airports, and it is important to use node symbols as anchors for airport locations. On the other hand, node symbols for visualizing commodity flows between regions such as states would be a reasonable choice for highlighting region-to-region flows that could originate anywhere from the origin region and end anywhere at the destination region. The user can hide point symbols by simply making the fill color transparent with no stroke-width for the outline of the symbols.

2.3. Choropleth map

FlowMapper supports generation of a choropleth map to support contextualizing of flow patterns with regional attributes and/or flow characteristics. Choropleth maps are especially important when flow data is from regions-to-regions such as state-to-state migration flows. Although the user may choose not to symbolize the choropleth map, it is often beneficial to use a choropleth map to illustrate a location-based measure of flows such as net-flow, net-flow ratio, or any other meaningful attribute such as population density.

2.4. Base map and map projections

Flow Mapper provides a variety of base maps produced by ESRI, Stamen and OpenStreetMap. For ESRI base maps, the user can choose to add reference labels for places. In addition, FlowMapper currently supports and provides the users the choice of following map projections: Albers Equal Area projection is available for U.S., Africa, Australia, China, Europe and South America. Robinson and Mercator projection are available for mapping flows at the world scale. More projections such as Equal Earth will be added in future revisions of FlowMapper. The user can also request a specific projection to be added on FlowMapper.

2.5. Computational tools

FlowMapper currently supports two functions for processing flow and location data under Tools menu: “Polygons to Points” and “Normalize Flows”. “Polygons to Points” tool transforms a polygon JSON file to a (node) point csv file using the geometric centroid of each polygon. Normalize flows function takes an input flow csv file and transforms it to a modularity flow file using an expectation calculation based on total flow volumes for each node. Flow normalization helps remove the effect of total flow or population size by calculating the difference between the actual flow and the expected volume of flow for each pair of locations (nodes). We explain the formula for transforming raw flows to modularity flows in the Appendices Section 2.

3. Map use cases

We demonstrate the utilities of Flow Mapper using three scenarios that portray distinct flow map data sets, map extent, scale, and layout.

3.1. Scenario 1: Banana trade (2019) within South America

Figure 3 illustrates banana trade flows between countries in South America from 2019 (FAO, 2021). We depict the net-flow ratio for each country on the choropleth map, the total flow volume (import + export) on the proportional symbol map, and volume of trade flows between countries using the teardrop symbol. The purple to green color scheme was selected as to represent the diverging netflow ratio from -1 to 1. Net-flow ratio highlights the countries in purple, with a negative netflow due to larger export of bananas to other countries in green with a positive netflow (more import than export). The black circles are the centroid for each country which represents the total flow. The flows were drawing using the teardrop style with the blue color scheme. This provides sufficient contrast between flows and regions that show the netflow ratio using the purple and green color scheme. Countries with a warmer climate that are closer to the equator export more bananas to countries in the south of the continent that have less suitable climate for producing bananas. From the nodes showing the total flow, we can see that Ecuador exports more bananas to other countries than Brazil and Peru while countries like Argentina and Chile are the major importers of bananas within South America.

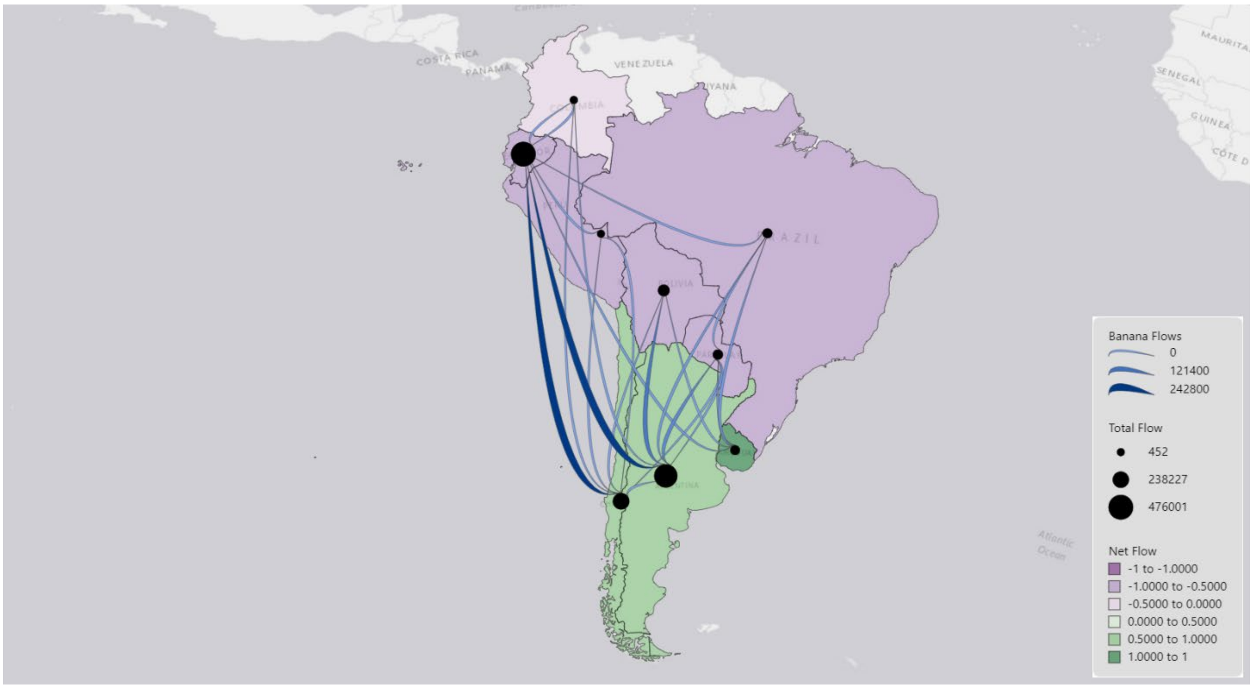


Figure 3: Banana trade flows between countries in South America

3.2. Scenario 2: Historical migration in the United States

Figure 4 illustrates the number of families migrated between states from 1887-1924 derived from a population-scale family tree data set (Koylu et al., 2020). The figure depicts the net flow for the polygon features with a diverging color scheme, from blue (more out-migrants than in-migrants) to red (more in-migrants than out-migrants). Proportional symbology was used to represent the points (nodes) for total flow as well as the flow symbols. The black points and grey lines had the greatest contrast among the blue-red polygon color scheme. As for interpreting the visualization, unsurprisingly, the predominant flows were East to West and between adjacent states. Particularly, Oklahoma appears to have a great influx of flows during this period, most likely because of the oil boom, with the first column of oil being discovered in 1897 (Aoghs.org Editors, 2020).

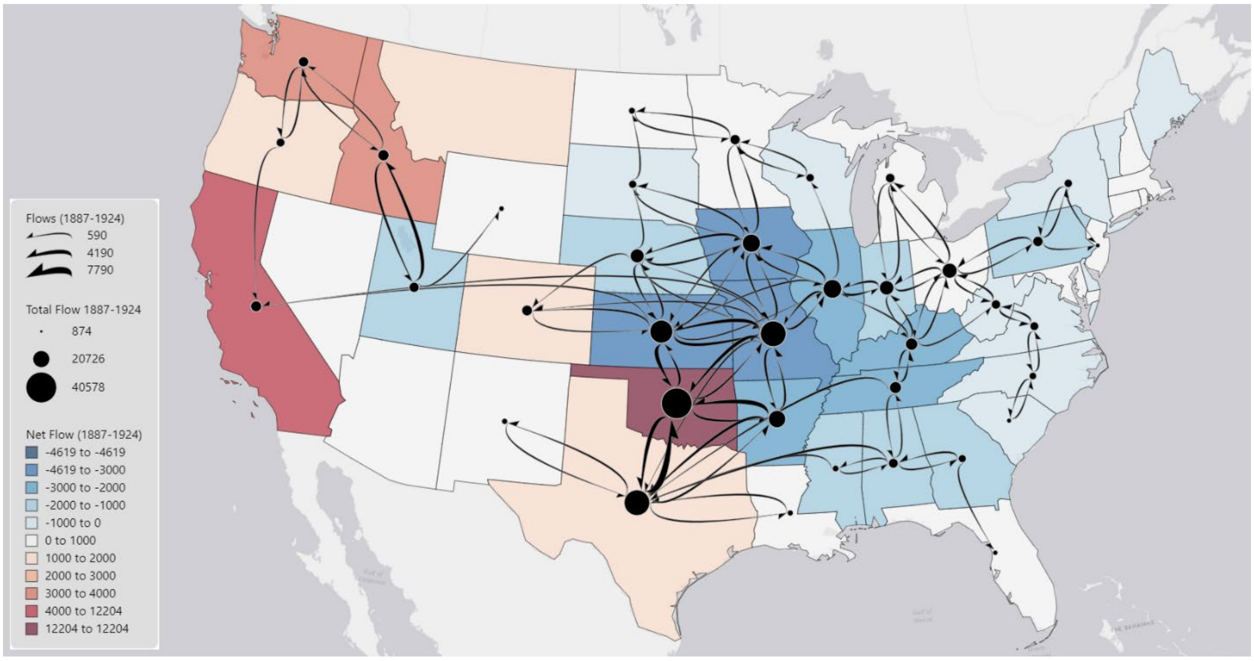


Figure 4: Historical migration of families between 1887 and 1924 in the United States

3.3. Scenario 3: Bike trips in Chicago, United States

Figure 5.a illustrates bike trips in Chicago throughout the month of April 2021 (Divvy, 2021). We use point symbols for bike stations and teardrop style for flows. We use netflow to distinguish how each bike station is used, whether as common origins where people pick up bikes or common destinations where people drop off bikes. The downtown area of Chicago has the highest density of flows. One of the most notable findings is that most of the flows are shorter in distance which means bikeshare users in Chicago generally stay in the neighborhood they begin the bike trip with or go to an adjacent neighborhood, which reflect the smaller clusters of flows in the map as opposed to flows that go across a great distance to other areas. In Figure 5.b, the bikeshare flows are zoomed in on the downtown area of Chicago to provided greater detail of the flow patterns. Different from Figure 5.a, Figure 5.b uses the curve symbol with half-arrow to reinforce the perception of directions, and a dark basemap with light and neon colors to provide increased contrast. In this map, the Lake Shore Dr. and Monroe St. bike station is selected to highlight flows from and to this station. The station is nearby attractions such as Maggie Daley Park, the Art Institute of Chicago, and Cloud Gate. The major in flows to this station come from the Chicago Field Museum, the Shedd Aquarium, and the Alder Planetarium from southern part of downtown. The largest number of bike trips for this station are from and to the Navy Pier bikeshare station which is to the northwest. Other larger out flows from Lake Shore Dr. and Monroe St. station are located along a bike path that goes along Lake Shore Dr. that is to the northwest of this location.

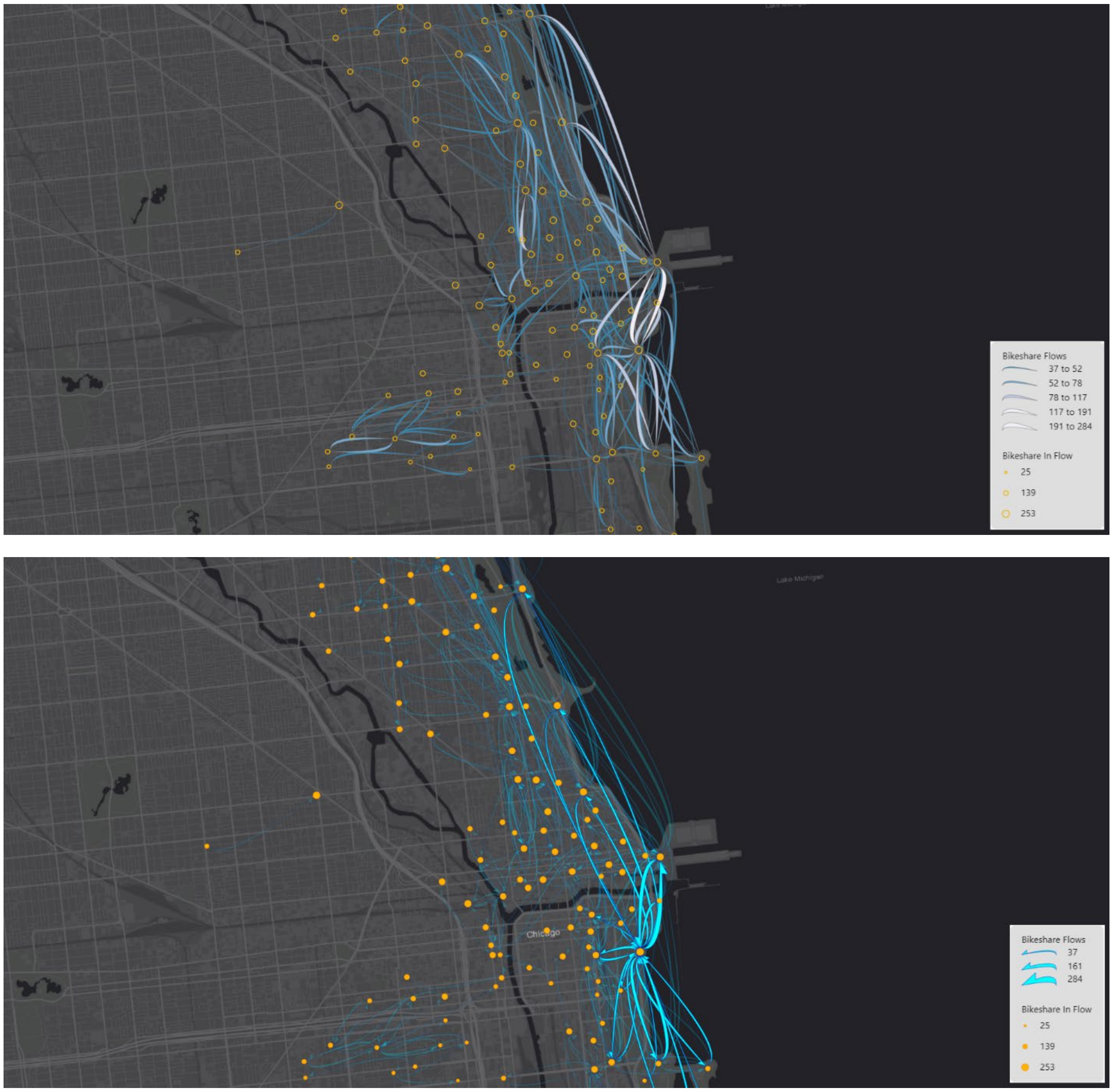


Figure 5: (a) Bike trips in Chicago between bike stations using teardrop symbol (b) Selected bike trips from and to Lake Shore Dr. and Monroe St. bike station using curve with half-arrow symbol.

4. Conclusions and future work

In this article, we summarize four main contributions of FlowMapper. First, FlowMapper which allows users to easily produce, customize and share flow maps by using their own data using a lightweight front-end design. We plan to expand the sharing of flow maps by enabling users to host and process their data and flow map designs on our back-end server and produce web links to user-generated flow maps. Second, we introduce the algorithms for drawing curved flow paths with varying thickness along a flow line, which reduces the visual cluttering and overlapping at the origin and destination locations. Alternative symbology choices allow the user to support different flow map reading tasks such as comparing flow magnitudes and directions and identifying flow and location clusters that are strongly connected with each other. In future work, we plan to implement more flow line styles that incorporate origin-destination coloring and partial lines, and flow line designs used by others (Jenny et al., 2017). We also plan to implement undirected and one-way directed flow maps. Third, FlowMapper supports additional layers to contextualize flow patterns with locational characteristics such as net-flow, total flow, or a relevant locational attribute. It supports user interactions to zoom, filter, and obtain details-on-demand functions to support visual information seeking. However, there is a need to guide design of OD flow maps using a comprehensive typology of patterns, tasks, and data. Following the advancements in deriving comprehensive pattern typology and cartographic principles for non-spatial graph and geographic movement and trajectory data (Dodge, 2019; Dodge, Weibel, & Lautenschutz, 2008; Purchase et al., 2008), we plan to support the design decisions for FlowMapper by developing a pattern typology for OD flow data and patterns. This will provide some guidance for users for identifying types of patterns with preconfigured symbology types. Finally, the web-based architecture of FlowMapper supports server-side computational capabilities to process and summarize large flow data to reveal natural patterns of flows. While we currently support flow normalization, we plan to implement flow data processing such as regionalization (Guo, 2008) and multi-scale flow mapping (Zhu et al., 2019), and also support space-time flow mapping (Andrienko et al., 2017; von Landesberger et al., 2016).

Software

The front end of FlowMapper is built with HTML/CSS/JavaScript for the website, OpenLayers for the mapping framework including the base maps and basic interactive map functions, D3 (Data-Driven Documents) (Bostock, Ogievetsky, & Heer, 2011) for the creation of interactive choropleth, point symbol and flow symbol vectors. FlowMapper utilizes a Java back-end design with connections with a PostgreSQL/PostGIS database with an Apache Tomcat server. Currently, flow data transformation and normalization features can be employed using the back-end server. Back-end design will enable future data hosting, processing and simplification using flow data processing such as regionalization (Guo, 2008) and multi-scale flow mapping (Zhu et al., 2019) capabilities.

Data availability statement

All the source code to produce both back-end and front-end development of FlowMapper and the example flow data sets used in this article are publicly available with a DOI at: https://doi.org/10.6084/m9.figshare.14593380.v2. The data sets used in this study do not contain human subjects, and they are in the public domain. Banana trade flow data between countries in South America from 2019 were derive from Food and Agriculture Organization of United Nations website: http://www.fao.org/faostat/en/#data/TM (FAO, 2021). The country polygons were obtained from Natural Earth Data (Natural Earth, 2021). Historical family migration data between states from 1887-1924 were derived from a population-scale family tree data set produced by Koylu *et al.* (2020). Bike trip data in Chicago throughout the month of April 2021 were derived from https://www.divvybikes.com/system-data (Divvy, 2021). All the source code and materials for FlowMapper are also published on this GitHub link: https://github.com/geo-social/flowmapper.

Acknowledgements

The authors thank the department of Geographical and Sustainability Sciences at the University of Iowa for supporting the development of FlowMapper as a teaching material to be used in geography, GIS, and other relevant courses. The authors thank Mert Erdemir, Beichen Tian and Hoeyun Kwon for their kind support and feedback.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Andrienko, G., Andrienko, N., Fuchs, G., & Wood, J. (2017). Revealing patterns and trends of mass mobility through spatial and temporal abstraction of origin-destination movement data. *IEEE transactions on visualization and computer graphics, 23*(9), 2120-2136.

Aoghs.org Editors. (2020). Oklahoma Oil History. Retrieved from https://aoghs.org/oil-almanac/oklahoma-oil-history/

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D³ data-driven documents. *IEEE transactions on visualization and computer graphics, 17*(12), 2301-2309.

Boyandin, I. (2021). flowmap.blue. Retrieved from https://flowmap.blue

Dent, B., Torguson, J., & Hodler, T. (2008). *Thematic map design*: McGraw-Hill New York, New York, NY.

Divvy. (2021). Historical bike trip data. Retrieved from https://www.divvybikes.com/system-data

Dodge, S. (2019). A Data Science Framework for Movement. *Geographical Analysis*.

Dodge, S., Weibel, R., & Lautenschutz, A.-K. (2008). Towards a taxonomy of movement patterns. *Inf Visualization, 7*(3-4), 240-252.

Dong, W., Wang, S., Chen, Y., & Meng, L. (2018). Using eye tracking to evaluate the usability of flow maps. *ISPRS International Journal of Geo-Information, 7*(7), 281.

FAO. (2021). Food and Agriculture Organization of United Nations (FAO). Retrieved from http://www.fao.org/faostat/en/#data/TM

Graser, A., Schmidt, J., Roth, F., & Brändle, N. (2019). Untangling origin-destination flows in geographic information systems. *Information Visualization, 18*(1), 153-172.

Guo, D. (2008). Regionalization with dynamically constrained agglomerative clustering and partitioning (REDCAP). *International Journal of Geographical Information Science, 22*(7), 801-823. doi:10.1080/13658810701674970

Guo, D. (2009). Flow Mapping and Multivariate Visualization of Large Spatial Interaction Data. In (Vol. 15, pp. 1041-1048). IEEE Transactions on Visualization and Computer Graphics: IEEE.

Guo, D., & Zhu, X. (2014). Origin-Destination Flow Data Smoothing and Mapping. *Visualization and Computer Graphics, IEEE Transactions on, PP*(99), 1-1. doi:10.1109/TVCG.2014.2346271

Holten, D., Isenberg, P., Fekete, J.-D., & Van Wijk, J. (2010). Performance Evaluation of Tapered, Curved, and Animated Directed-Edge Representations in Node-Link Graphs.

Holten, D., & van Wijk, J. J. (2009). Force-Directed Edge Bundling for Graph Visualization. *Computer Graphics Forum, 28*(3), 983-990.

Jenny, B., Stephen, D. M., Muehlenhaus, I., Marston, B. E., Sharma, R., Zhang, E., & Jenny, H. (2017). Force-directed layout of origin-destination flow maps. *International Journal of Geographical Information Science, 31*(8), 1521-1540.

Jenny, B., Stephen, D. M., Muehlenhaus, I., Marston, B. E., Sharma, R., Zhang, E., & Jenny, H. (2018). Design principles for origin-destination flow maps. *Cartography and Geographic Information Science, 45*(1), 62-75.

Koylu, C. (2014). *Understanding Geo-Social Network Patterns: Computation, Visualization, and Usability.* (Ph.D.). University of South Carolina,

Koylu, C., & Guo, D. (2017). Design and evaluation of line symbolizations for origin–destination flow maps. *Information Visualization, 16*(4), 309-331. doi:10.1177/1473871616681375

Koylu, C., Guo, D., Huang, Y., Kasakoff, A., & Grieve, J. (2020). Connecting family trees to construct a population-scale and longitudinal geo-social network for the U.S. *International Journal of Geographical Information Science*. doi:10.1080/13658816.2020.1821885

Lan, Y., Delmelle, E., & Delmelle, E. (2021). NDS: an interactive, web-based system to visualize urban neighborhood dynamics in United States. *Journal of Maps, 17*(1), 62-70. doi:10.1080/17445647.2021.1911867

Natural Earth. (2021). Country Shapefiles. Retrieved from https://www.naturalearthdata.com

Nost, E., Rosenfeld, H., Vincent, K., Moore, S. A., & Roth, R. E. (2017). HazMatMapper: an online and interactive geographic visualization tool for exploring transnational flows of hazardous waste and environmental justice. *Journal of Maps, 13*(1), 14-23. doi:10.1080/17445647.2017.1282384

Parisi, T. (2012). *WebGL: up and running*: " O'Reilly Media, Inc.".

Phan, D., Xiao, L., Yeh, R., & Hanrahan, P. (2005). *Flow map layout.* Paper presented at the IEEE Symposium on Information Visualization.

Purchase, H., Andrienko, N., Jankun-Kelly, T., & Ward, M. (2008). Theoretical Foundations of Information Visualization. In A. Kerren, J. Stasko, J.-D. Fekete, & C. North (Eds.), *Information Visualization* (Vol. 4950, pp. 46-64): Springer Berlin / Heidelberg.

Slocum, T. A., McMaster, R. B., Kessler, F. C., & Howard, H. H. (2009). *Thematic cartography and geovisualization*: Pearson Prentice Hall Upper Saddle River, NJ.

Tao, R., & Thill, J. C. (2016). Spatial cluster detection in spatial flow data. *Geographical Analysis, 48*(4), 355-372.

Tobler, W. R. (1987). Experiments in migration mapping by computer. *American Cartographer, 14*, 155–163.

von Landesberger, T., Brodkorb, F., Roskosch, P., Andrienko, N., Andrienko, G., & Kerren, A. (2016). Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE transactions on visualization and computer graphics, 22*(1), 11-20.

Ware, C., Kelley, J. G., & Pilar, D. (2014). Improving the Display of Wind Patterns and Ocean Currents. *Bulletin of the American Meteorological Society, 95*(10), 1573-1581.

Zhu, X., Guo, D., Koylu, C., & Chen, C. (2019). Density-based multi-scale flow mapping and generalization. *Computers, Environment and Urban Systems, 77*, 101359. doi:10.1016/j.compenvurbsys.2019.101359

Figure captions and alt text

Figure 1: Flow line styles (a) Curve with half-arrow (b) Straight with half-arrow (c) Tapered (d) Teardrop

Figure 1 Alt Text: Four flow symbol design FlowMapper currently supports: Curve with half-arrow, tapered curve, teardrop curve, and straight-line with half-arrow. All the curves are asymmetric to enhance readability of directions in two-way flow maps.

Figure 2: Scaling and legend alternatives for flow symbols (a) a single color (black) proportional flow line symbol, which only scales line thickness to be proportional to flow volume. (b) Flow symbol with both the color and thickness are scaled proportional to flow volume (b) a classified flow legend which uses five classes assigned by a quantile classification.

Figure 2 Alt Text: Scaling and legend alternatives for flow symbols. Figure 2.a illustrates a proportional flow symbol legend with a single color (black), while Figure 2.b illustrates a flow symbol legend with a continuous color scheme from white to blue. Figure 2.c illustrates a classified flow magnitude using a quantile classification with 5 classes using a blue color scheme. Classification is applied to both flow colors and widths.

Figure 3: Banana trade flows between countries in South America

Figure 3 Alt Text: Banana trade flows between countries in South America from 2019 (FAO, 2021). We depict the net-flow ratio for each country on the choropleth map, the total flow volume (import + export) on the proportional symbol map, and volume of trade flows between countries using the teardrop style. Countries with a warmer climate that are closer to the equator export more bananas to countries in the south of the continent that have less suitable climate for producing bananas.

Figure 4: Historical migration of families between 1887 and 1924 in the United States

Figure 4 Alt Text: The number of families migrated between states from 1887-1924 derived from a population-scale family tree data set (Koylu et al., 2020). The predominant flows were East to West and between adjacent states. Particularly, Oklahoma appears to have a great influx of flows during this period, most likely because of the oil boom, with the first column of oil being discovered in 1897 (Aoghs.org Editors, 2020).

Figure 5: (a) Bike trips in Chicago between bike stations using teardrop symbol (b) Selected bike trips from and to Lake Shore Dr. and Monroe St. bike station using curve with half-arrow symbol.

Figure 5 Alt Text: Bike trips in Chicago in April 2021 (Divvy, 2021). Figure 5.a uses point symbols for bike stations, and teardrop flow symbol for flows. Figure 5.b uses the curve symbol with half-arrow to reinforce the perception of directions, and a dark basemap with light and neon colors to provide increased contrast. In this map, the Lake Shore Dr. and Monroe St. bike station is selected to highlight flows from and to this station. The largest number of bike trips for this station are from and to the Navy Pier bikeshare station which is to the northwest.

Appendices

1. Flow path symbology and algorithms

Here we illustrate the asymmetric Bezier curve with a half-arrow connecting an origin node to a destination node (Figure A.1). The flow symbol is curvy at the origin and straight at the destination end to increase readability of flow maps by minimizing the flow lines that are converging to and diverging from nodes. Curved flow line with half-arrow symbology consists of two cubic Bezier curves between the origin and destination points with an additional half-arrow drawn at the destination. We paint the area between the two curves and the half-arrow to generate the flow line style. We calculate the coordinate points explained below using the algorithm described in Appendix Table A.1. There are four major steps in curve with half-arrow algorithm:

1- We first identify the origin and destination points of the flow symbol on the periphery of the node symbols (circles) instead of directly drawing the flow symbol between the coordinates (centroids) of the origin-destination points. We name these points as P0 (origin) and P3 (destination), which are at the intersection of the periphery of the node symbol (circle) and the imaginary straight line between P0 and P3.
   We then draw a straight line from P0 to an offset point (P1) which is perpendicular to the imaginary straight line between P0 and P3. Because curves converge at the destination end touching the circle of the node symbol, the offset point allows the flows to be drawn away from the origin node and reduce cluttering of potential overlap between the flows that leave the node and flows that arrive at the node.

2- We draw the first Bezier cubic curve that will produce the outer edge of the flow symbol. A cubic Bezier curve is drawn with three points. While the first two points are control points, the last point is the end point for the curve. To draw a half-arrow, we estimate the end point (P2) for the outer curve to be before the destination point (P3) on an imaginary Bezier curve line to the destination point (P3). To draw the asymmetric curve, we now have the start point (P1), the two control points: CP1_C1 (control point 1 for curve 1) and CP2_C1 (control point 2 for curve 1) that are marked with a red stroke color, and the end point (P2).

3- We draw the half-arrow using two straight lines. The first line is the straight line between P2 and EP3 (elbow point for the arrow), and the second line is the destination point (P3). Now, the outer edge of the flow symbol is drawn.

4- Drawing of the inner flow symbol is straightforward since it does not include an offset or a half-arrow. The inner flow is also a cubic Bezier curve that starts from P3, uses two control points of CP1_C2 and CP2_C2, and the end point P0.

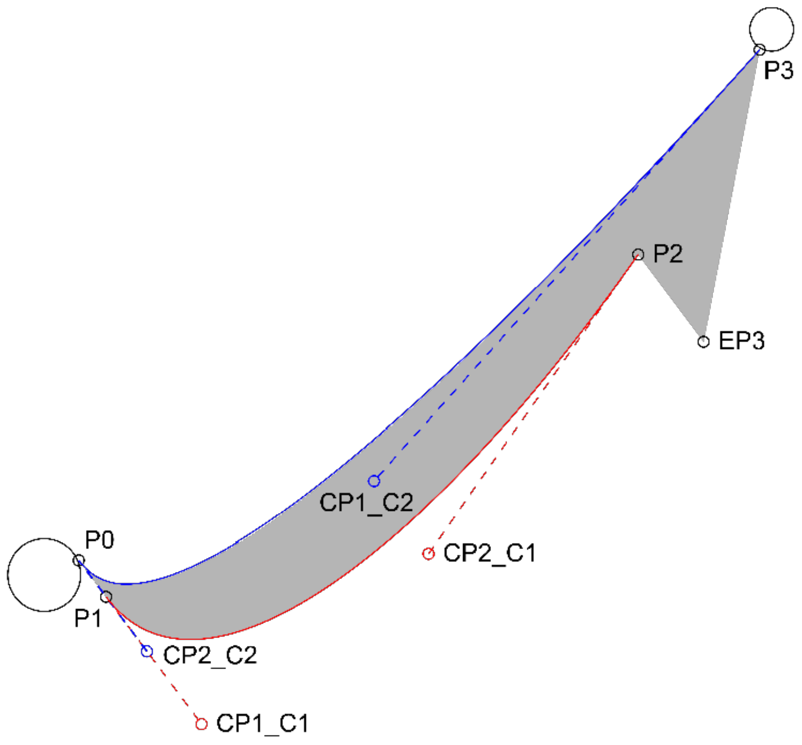


Figure A.1: Bezier curve with half-arrow. P0 and P3 are origin and destination points. CP points are control points for the inner (blue) and outer (red) Bezier curves. P1 and P2 are offset points for origin and destination, respectively. EP3 is the elbow point for drawing the half-arrow.

In tapered flow line symbol, the only difference is in the drawing of the outer Bezier curve. We simply remove the destination offset point (P2) and the elbow point (EP) and draw a cubic curve from P1 to P3 using the same control points P1_CP1 and new control point calculated directly from the destination point P3. Teardrop flow symbol is the reverse of the tapered symbol. In teardrop design, the origin and destination points are flipped, and the origin offset point P1 (which now would be a destination offset point) is removed. Teardrop has been found to be effective for direction tasks in flow map reading (Koylu & Guo, 2017; Ware et al., 2014). Straight flow line symbol includes a destination offset and an elbow point to draw the half-arrow. Straight flow with half-arrow has been used extensively in flow mapping (Boyandin, 2021; Guo, 2009).

Below is the JavaScript code for producing half-arrow Bezier curve flow line symbolization.

Table A.1: The algorithm to construct Bezier curve with half-arrow.

Input: x0, y0: x and y coordinates of the origin location
x3, y3: x and y coordinates of the destination location
flowSize: Flow thickness (in pixel size) determined by the flow magnitude (e.g., volume)
sourceRadius: Size of origin symbol (circle)
targetRadius: Size of destination symbol (circle)
righthandrule: if true then draw flows using the right-hand traffic rule, if false the draw using the left-hand traffic rule

Output: Curved path with half-arrow
Corresponding points calculated for Figure 2:
P1: xc1, yc1
P2: xc2, yc2
CP1_C1:xc13rd, yc13rd
CP2_C1: xc23rd, yc23rd
EP3: x3elbow, y3elbow
CP1_C2: x23rd, y23rd
CP2_C2: x13rd, y13rd

```
function drawCurve (x0, y0, x3, y3, flowSize, sourceRadius, targetRadius, righthandrule){

        var arrowlen = 2.42;
        var arrowwidthconstant = flowSize * 1.1;

        // adjusting half-arrow size
        if(flowSize < 10 && flowSize >= 8){
              arrowlen = 2.64;
              arrowwidthconstant = flowSize * 1.2;
        }
        else if(flowSize < 8 && flowSize >= 6){
              arrowlen = 3.08;
              arrowwidthconstant = flowSize * 1.4;
        }
        else if(flowSize < 6 && flowSize >= 4){
              arrowlen = 4.4;
              arrowwidthconstant = flowSize * 2;
        }
        else if(flowSize < 4 && flowSize >= 3){
              arrowlen = 6.6;
              arrowwidthconstant = flowSize * 3;
        }
        else if(flowSize < 3 && flowSize >= 2){
              arrowlen = 8.8;
              arrowwidthconstant = flowSize * 4;
        }
        else if(flowSize < 2){
              arrowlen = 11;
              arrowwidthconstant = flowSize * 5;
        }

        var ndsize0 = sourceRadius;
        var ndsize3 = targetRadius;
        var dx = Math.abs(x3 - x0);
        var dy = Math.abs(y3 - y0);

        var len = Math.sqrt(dx * dx + dy * dy);
        if (len < (ndsize0 + ndsize3) * 1.2)
              return;

        // Shorten the length so that the line only touches the node circle
        var haselbow = true;
        if (haselbow) {
              x0 = x0 + (x3 - x0) * ndsize0 / len;
              y0 = y0 + (y3 - y0) * ndsize0 / len;
              x3 = x3 - (x3 - x0) * ndsize3 / (len - ndsize0);
              y3 = y3 - (y3 - y0) * ndsize3 / (len - ndsize0);
        }

        // Four corners of a flow line
        var xc1 = null, yc1 = null, xc2 = null, yc2 = null;
        // Four points to round the head
        var x3elbow1 = null, y3elbow1 = null;
        var sign = -1;
        var xdelta = null, ydelta = null;
        xarrowdelta = null, yarrowdelta = null, xgap = null, ygap = null;
        var gap = flowSize * 0.05;
        if (y0 == y3) { // horizontal
              xdelta = 0;
              ydelta = flowSize / 2;
              xarrowdelta = 0;
              yarrowdelta = arrowwidthconstant / 1.0;
              xgap = 0;
              ygap = gap;
        } else if (x0 == x3) { // vertical
              ydelta = 0;
              xdelta = flowSize / 2;
              yarrowdelta = 0;
              xarrowdelta = arrowwidthconstant / 1.0;
              xgap = gap;
              ygap = 0;
        } else {
              var v = (x3 - x0) / (y0 - y3);
              xdelta = flowSize / 2.0 /  Math.sqrt(1 + v * v);
              ydelta = Math.abs(xdelta * v);
              xarrowdelta = arrowwidthconstant / Math.sqrt(1 + v * v);
              yarrowdelta = Math.abs(xarrowdelta * v);
              xgap = gap / Math.sqrt(1 + v * v);
              ygap = Math.abs(xgap * v);
              if (v < 0)
                    sign = 1;
        }
        x0 = (y0 > y3) ? x0 + xgap : x0 - xgap;
        x3 = (y0 > y3) ? x3 + xgap : x3 - xgap;
        y0 = (x0 > x3) ? y0 - ygap : y0 + ygap;
        y3 = (x0 > x3) ? y3 - ygap : y3 + ygap;

        if(righthandrule)
        {
              xc1 = (y0 > y3) ? x0 + xdelta / 2 : x0 - xdelta / 2;
              yc1 = (x0 > x3) ? y0 - ydelta / 2 : y0 + ydelta / 2;

        yc2 = (x0 > x3) ? y3 - ydelta + arrowlen * xdelta * sign : y3 + ydelta - 
  arrowlen * xdelta * sign;
        xc2 = (y0 > y3) ? x3 + xdelta + arrowlen * ydelta * sign : x3 - xdelta - 
  arrowlen * ydelta * sign;

        x3elbow1 = (y0 > y3) ? x3 + xarrowdelta + arrowlen * ydelta * sign : x3 
  - xarrowdelta - arrowlen * ydelta * sign;
        y3elbow1 = (x0 > x3) ? y3 - yarrowdelta + arrowlen * xdelta * sign : y3 
  + yarrowdelta - arrowlen * xdelta * sign;

        var arcxdelta = xdelta * len / 4 / flowSize;
        var arcydelta = ydelta * len / 4 / flowSize;
        var x13rd = (y0 > y3) ? x0 + arcxdelta : x0 - arcxdelta;
        var y13rd = (x0 > x3) ? y0 - arcydelta : y0 + arcydelta;
        var x23rd = (y0 > y3) ? x0 + (x3 - x0) / 3 + arcxdelta : 
  x0 + (x3 - x0) / 3 - arcxdelta;
        var y23rd = (x0 > x3) ? y0 + (y3 - y0) / 3 - arcydelta : 
  y0 + (y3 - y0) / 3 + arcydelta;
        arcxdelta = arcxdelta + xdelta;
        arcydelta = arcydelta + ydelta;
        var xc13rd = (y0 > y3) ? x0 + arcxdelta : x0 - arcxdelta;
        var yc13rd = (x0 > x3) ? y0 - arcydelta : y0 + arcydelta;
        var xc23rd = (y0 > y3) ? x0 + (x3 - x0) / 3 + arcxdelta : x0 + (x3 - x0) 
  / 3 - arcxdelta;
        var yc23rd = (x0 > x3) ? y0 + (y3 - y0) / 3 - arcydelta : y0 + (y3 - y0) 
  / 3 + arcydelta;

        }else
        {
        //left-hand traffic rule
        xc1 = (y0 < y3) ? x0 + xdelta / 2 : x0 - xdelta / 2;
        yc1 = (x0 < x3) ? y0 - ydelta / 2 : y0 + ydelta / 2;
        yc2 = (x0 < x3) ? y3 - ydelta + 2.5 * xdelta * sign : 
  y3 + ydelta - 2.5 * xdelta * sign;
        xc2 = (y0 < y3) ? x3 + xdelta + 2.5 * ydelta * sign : 
  x3 - xdelta - 2.5 * ydelta * sign;

        x3elbow1 = (y0 < y3) ? x3 + xarrowdelta + 2.5 * ydelta * sign : 
  x3 - xarrowdelta - 2.5 * ydelta * sign;
        y3elbow1 = (x0 < x3) ? y3 - yarrowdelta + 2.5 * xdelta * sign : 
  y3 + yarrowdelta - 2.5 * xdelta * sign;

        var arcxdelta = xdelta * len / 4 / flowSize;
        var arcydelta = ydelta * len / 4 / flowSize;
        var x13rd = (y0 < y3) ? x0 + arcxdelta : x0 - arcxdelta;
        var y13rd = (x0 < x3) ? y0 - arcydelta : y0 + arcydelta;
        var x23rd = (y0 < y3) ? x0 + (x3 - x0) / 3 + arcxdelta : 
  x0 + (x3 - x0) / 3 - arcxdelta;
```

```
        var y23rd = (x0 < x3) ? y0 + (y3 - y0) / 3 - arcydelta :
y0 + (y3 - y0) / 3 + arcydelta;
        arcxdelta = arcxdelta + xdelta;
        arcydelta = arcydelta + ydelta;

        var xc13rd = (y0 < y3) ? x0 + arcxdelta : x0 - arcxdelta;
        var yc13rd = (x0 < x3) ? y0 - arcydelta : y0 + arcydelta;
        var xc23rd = (y0 < y3) ? x0 + (x3 - x0) / 3 + arcxdelta : x0 + (x3 - x0)
/ 3 - arcxdelta;
        var yc23rd = (x0 < x3) ? y0 + (y3 - y0) / 3 - arcydelta : y0 + (y3 - y0)
/ 3 + arcydelta;
        }

        return "M" + x0 + "," + y0
+ " L" + xc1 + "," + yc1
+ " C" + xc13rd + ","+ yc13rd + " " +xc23rd+ "," +yc23rd+ " "+ xc2 + "," + yc2
+ " L" + x3elbow1 + "," + y3elbow1
+ " L" + x3 + "," + y3
+ " C" + x23rd + "," +y23rd+ " " +x13rd+ "," +y13rd+ " " + x0 + "," + y0;
}
```

2. Flow normalization

We use an adjusted flow volume formula to calculate the expected number of flows between locations.

$$\text{Expected Flow (O, D)} = F_O\, F_D\, f\,(O, D) \,/\, (F_S{}^2 - \textstyle\sum_{i=0}^{n} F_i{}^2\,)$$

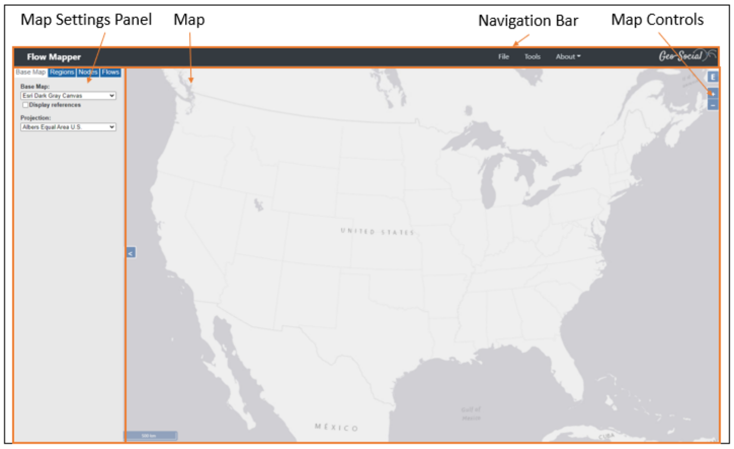
where EF (O, D) is the expected number of flows between origin O and destination D, F_O is the observed number of flows between county O and its connections, F_D is the observed number of flows between county D and its connections, f (O, D) is the observed number of flows between county O and county D, F_S is the observed number of flows between all locations, and $\sum_{i=0}^{n} F_i{}^2$ is used to remove within-location expectations if there are any. Finally, modularity of a flow O-D is calculated as:

$$\text{MOD (O, D)} = \text{Observed Flow} - \text{Expected Flow}$$

where OF is actual (observed) number of flows, and EP is expected number of flows on link O-D. Using this formula, the raw counts of flows are transformed into a modularity graph, in which the weight of a link represents the modularity between two locations. If modularity value is positive the flow is above expectation, if the value is negative the flow is below expectation.

3. User interface and experience

We describe the user interface (UI) and the user experience (UX) of FlowMapper from a general user's perspective. Upon accessing FlowMapper, the user will see a base map in the center of the application, a navigation bar at the top, and a flow map settings panel on the left of the interface that is collapsible to provide more space for the map interface (Figure A.2.a). The map is where the flow dataset will be visualized after it is uploaded, and the flow map settings are given. Using the map, the user can pan around, zoom in on areas, and zoom out. To change the base map, the user will use the map settings panel under the 'Base Map' tab and select the desired base map from the Base Map drop-down menu. Within the 'Base Map' tab, the user can also change the map's projection using the drop-down 'Projection' menu (Figure A.2.b).



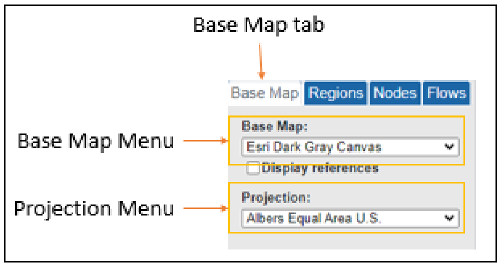
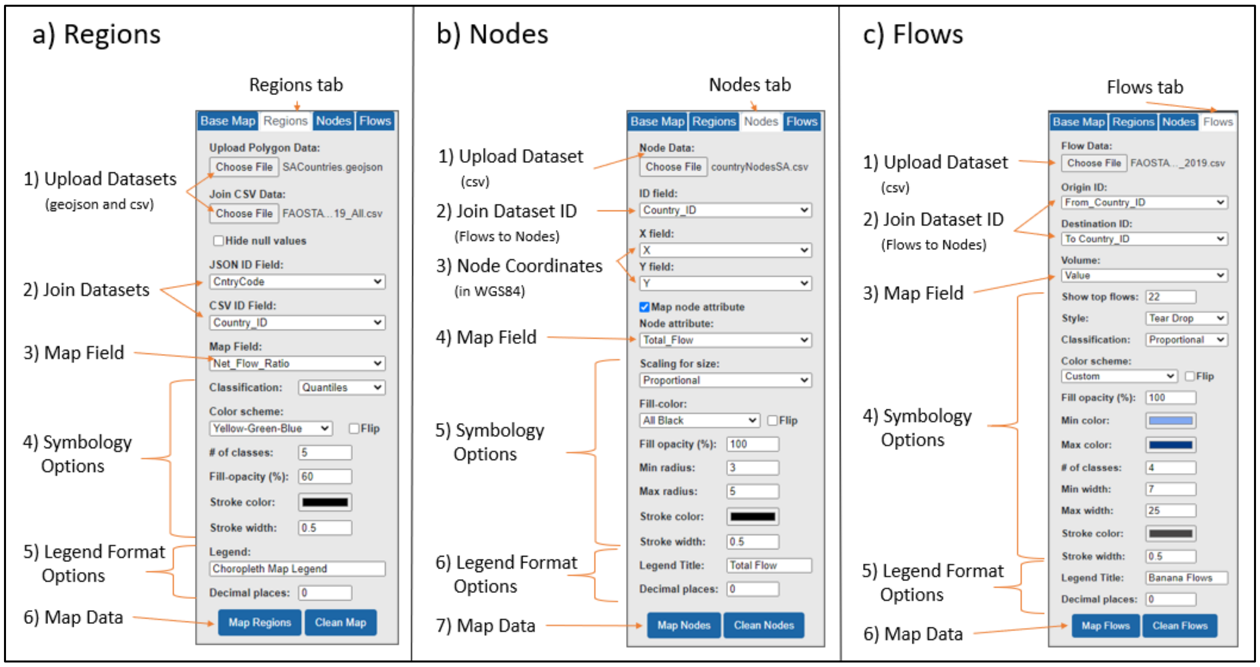


Figure A.2: (a) FlowMapper user interface and user experience and (b) base map controls

To start uploading flow data, the user can use regions (polygons), nodes (points), and flows (flow lines) to visualize their dataset. While regions are optional, nodes and flows are required for the flow map application. The regions are uploaded through the ‘Regions’ tab of the map settings panel. The regions dataset must be in a geojson format and the polygons within it should have an ID that joins to a csv dataset to map any relevant attribute data. Figure A.3.a shows an example dataset of a geojson uploaded for the polygon dataset and a csv dataset uploaded for the attribute dataset. Below the uploaded datasets are where the joins occur. Here the user would join the geojson ID to the csv ID. Once they are joined, the user can select a field to display the attribute data and then choose symbology options below in the panel. The symbology options are the data classification method, the color scheme, the number of classes for the classification (quantiles, equal intervals, natural breaks, linear scale, and manual breaks), fil-opacity (%), stroke color, stroke width. Once these options are filled out, the legend options can be formatted, such as the data layer title and the number of decimal places that should be displayed within the legend. Lastly, to add the dataset to the map with the symbology and legend options set, the user would select the lower left button “Map Regions”.

Flow Mapper requires users to upload a node (point) file to determine the origin and destination point coordinates for flows. The nodes data set should be uploaded from within the ‘Nodes’ tab as a csv. One can create point (node) data using Polygons to Points tool under “Tools” menu, which generates a point data set from the centroid of a polygon data set (regions) in JSON format. The data set should include a node ID which joins on a one-to-many relationship to the origin ID and destination ID of the flows csv dataset. The nodes dataset must also contain an “X” coordinate field and a “Y” coordinate field in latitude and longitude. The flow volume can be mapped and then symbolized with the same options that the regions tab provides. Like within the Regions tab, the legend can be formatted by title and number of decimal places. To create to the nodes on the map, the user would select “Map Nodes” at the bottom left of the nodes tab within the panel (Figure A.3.b).

To map the flows, the user should select the ‘Flows” tab within the Map Settings panel. The flow dataset should be uploaded as a csv, and then the user can input the fields that should be joined to the nodes, the flow origin ID and the flow destination ID. Next, the user would input the flow volume field and select symbology options. The symbology options are similar to those discussed from within the ‘Regions’ tab, except that the flows can also be visualized to show the top *n* number of flows for filtering purposes and the style of flow can be chosen. The style of flow can either curved half arrow, straight half arrow, tapered, or teardrop (Figure 1). Lastly, the user would input the legend formatting options and then select 'Map Flows' at the bottom left of the 'Flows' tab (Figure A.3.c).



Upload Polygon Data: SACountries.geojson
Join CSV Data: FAOSTA...19_All.csv
☐ Hide null values
JSON ID Field: CntryCode
CSV ID Field: Country_ID
Map Field: Net_Flow_Ratio
Classification: Quantiles
Color scheme: Yellow-Green-Blue ☐ Flip
# of classes: 5
Fill-opacity (%): 60
Stroke color: ⬛
Stroke width: 0.5
Legend: Choropleth Map Legend
Decimal places: 0

Node Data: countryNodesSA.csv
ID field: Country_ID
X field: X
Y field: Y
☑ Map node attribute
Node attribute: Total_Flow
Scaling for size: Proportional
Fill-color: All Black ☐ Flip
Fill opacity (%): 100
Min radius: 3
Max radius: 5
Stroke color: ⬛
Stroke width: 0.5
Legend Title: Total Flow
Decimal places: 0

Flow Data: FAOSTA..._2019.csv
Origin ID: From_Country_ID
Destination ID: To Country_ID
Volume: Value
Show top flows: 22
Style: Tear Drop
Classification: Proportional
Color scheme: Custom ☐ Flip
Fill opacity (%): 100
Min color: 🟦
Max color: 🟦
# of classes: 4
Min width: 7
Max width: 25
Stroke color: ⬛
Stroke width: 0.5
Legend Title: Banana Flows
Decimal places: 0

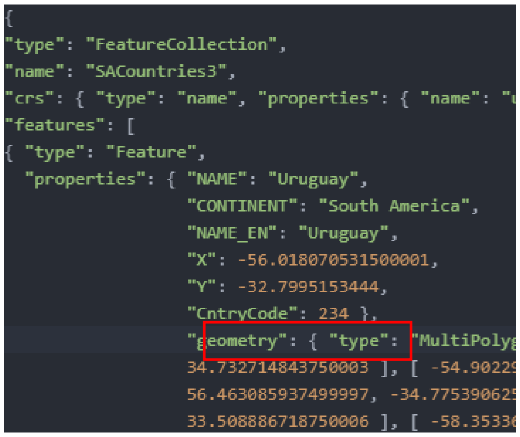
Figure A.3: Map Settings panel interface showing the Regions tab, the Nodes tab, and the Flows tab.

If the user wishes to save their current project, they should use the 'File' menu → 'Project' → 'Save Project'. The project will be presented as a json file which will include the joins, the classification or scaling, the symbology, legend settings, and geographic data. The project file will be downloaded locally. If the user wants to open their project in FlowMapper, the user should upload the json file by selecting 'File' menu → 'Project' → 'Load Project'. To export the map as an image, the user can go to the 'File' menu → 'Export' → and select 'Export SVG' to save it as an SVG, or 'Export PNG' to save it as a PNG.

In the navigation menu, the 'About' section provides help and guidance about the user interface and usability of FlowMapper. 'Project Help' gives the user general tips for using the interface, 'Developer Info' shows the Geo-Social Lab's contact information for making suggestions or receiving help on projects. The 'Tutorials' section within the 'About' menu provides the user with video tutorials about using FlowMapper, hoping to resolve users' questions and guide them through using the interface so they can more easily make their own flow map.

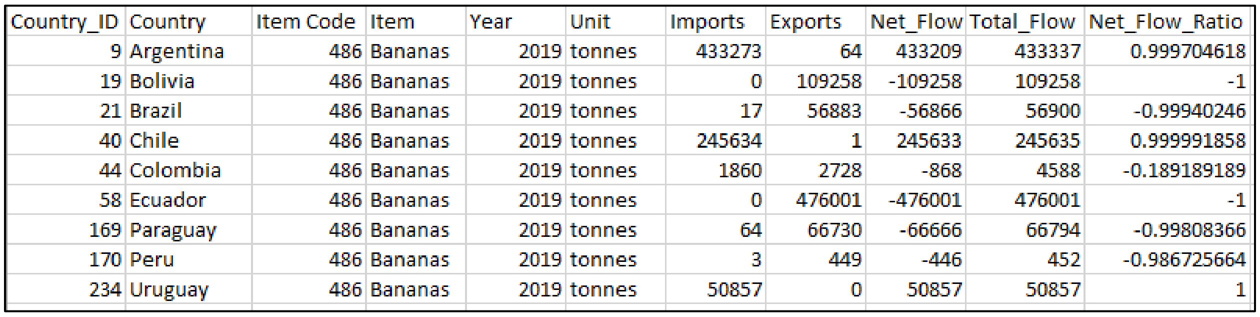
4. Data format

The datasets that can be loaded into FlowMapper must be in a certain format. First, the polygon dataset for the Choropleth map must be in a geojson or topojson format. For the polygon attribute data, a csv can be joined to the geojson within FlowMapper if both datasets have a common ID to join on. Second, the nodes dataset must be a csv file with X and Y coordinates in WGS84 datum. It must also have a common ID field so the flows can join to the nodes once the flow dataset is also uploaded. The nodes dataset should also have a volume field within it for the node symbology to be displayed within the map. Lastly, FlowMapper requires a flow dataset that should be in csv format. It should have an origin ID and a destination ID. Both ID's should match the common ID in the node csv file. Next, to represent the flow volume, the flow csv file should include a volume field which can represent the total number of flows from and to certain areas. Using an example dataset, Figure A4 displays the geojson for the polygon features. Outlined in red is the ID, or Country Code, that the attribute data within the csv (Figure A.5) will be joined on. Figure A.6 displays the node csv format with the X, Y, Country_ID, and Total_Volume fields. Figure A.7 shows the flow dataset where the "To Country Code" field and the "From Country Code" field would join to the node dataset to create the flows. The "Value" field represents the quantity of bananas being import or exported, which is equal to the flow volume filed needed for visualization. Once the dataset files are uploaded to FlowMapper, the user can save a project as one json file. If the user closes out of FlowMapper or the page needs to be refreshed, the user can always upload the json project file that will include the datasets settings and flow map visualization settings.



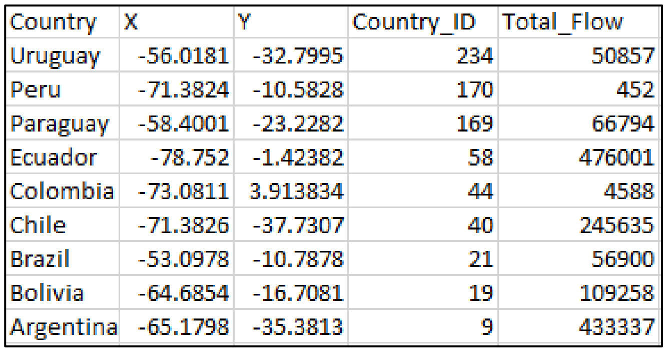
```
{
"type": "FeatureCollection",
"name": "SACountries3",
"crs": { "type": "name", "properties": { "name": "u
"features": [
{ "type": "Feature",
  "properties": { "NAME": "Uruguay",
                  "CONTINENT": "South America",
                  "NAME_EN": "Uruguay",
                  "X": -56.018070531500001,
                  "Y": -32.7995153444,
                  "CntryCode": 234 },
                  "geometry": { "type": "MultiPolyg
                  34.732714843750003 ], [ -54.90229
                  56.463085937499997, -34.77539062
                  33.508886718750006 ], [ -58.35336
```

Figure A.4: Example polygon geojson file with an id attribute to join with a csv file for creating a choropleth map.



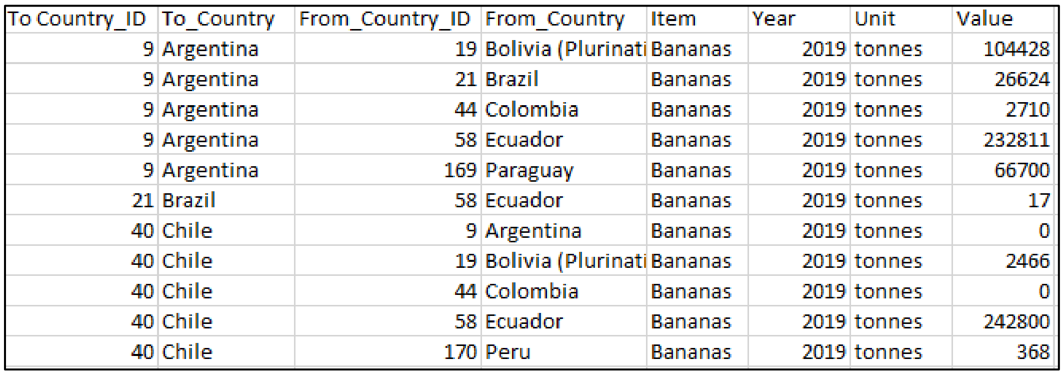
Country_ID	Country	Item Code	Item	Year	Unit	Imports	Exports	Net_Flow	Total_Flow	Net_Flow_Ratio
9	Argentina	486	Bananas	2019	tonnes	433273	64	433209	433337	0.999704618
19	Bolivia	486	Bananas	2019	tonnes	0	109258	-109258	109258	-1
21	Brazil	486	Bananas	2019	tonnes	17	56883	-56866	56900	-0.99940246
40	Chile	486	Bananas	2019	tonnes	245634	1	245633	245635	0.999991858
44	Colombia	486	Bananas	2019	tonnes	1860	2728	-868	4588	-0.189189189
58	Ecuador	486	Bananas	2019	tonnes	0	476001	-476001	476001	-1
169	Paraguay	486	Bananas	2019	tonnes	64	66730	-66666	66794	-0.99808366
170	Peru	486	Bananas	2019	tonnes	3	449	-446	452	-0.986725664
234	Uruguay	486	Bananas	2019	tonnes	50857	0	50857	50857	1

Figure A.5: A snapshot of a polygon csv file that contain attributes to create a choropleth map.



Country	X	Y	Country_ID	Total_Flow
Uruguay	-56.0181	-32.7995	234	50857
Peru	-71.3824	-10.5828	170	452
Paraguay	-58.4001	-23.2282	169	66794
Ecuador	-78.752	-1.42382	58	476001
Colombia	-73.0811	3.913834	44	4588
Chile	-71.3826	-37.7307	40	245635
Brazil	-53.0978	-10.7878	21	56900
Bolivia	-64.6854	-16.7081	19	109258
Argentina	-65.1798	-35.3813	9	433337

Figure A.6: A node csv file that contains the latitude and longitude coordinates of origin and destination locations, and the id field to match with the flow csv file.



To Country_ID	To_Country	From_Country_ID	From_Country	Item	Year	Unit	Value
9	Argentina	19	Bolivia (Plurinati	Bananas	2019	tonnes	104428
9	Argentina	21	Brazil	Bananas	2019	tonnes	26624
9	Argentina	44	Colombia	Bananas	2019	tonnes	2710
9	Argentina	58	Ecuador	Bananas	2019	tonnes	232811
9	Argentina	169	Paraguay	Bananas	2019	tonnes	66700
21	Brazil	58	Ecuador	Bananas	2019	tonnes	17
40	Chile	9	Argentina	Bananas	2019	tonnes	0
40	Chile	19	Bolivia (Plurinati	Bananas	2019	tonnes	2466
40	Chile	44	Colombia	Bananas	2019	tonnes	0
40	Chile	58	Ecuador	Bananas	2019	tonnes	242800
40	Chile	170	Peru	Bananas	2019	tonnes	368

Figure A.7: A flow csv file that contains origin, destination, and flow volume fields.