GoMA: A Grammar of Semantic Annotations on Maps

Hanning Shao, Haoyu Xu, and Xiaoru Yuan

Abstract—Maps, as one of the most effective means of conveying spatial information, are widely used in various scenarios. By overlaying rich visual elements onto cartographic maps, people further enhance the ability of maps to encode custom data. This includes thematic maps and map visualizations by experts, as well as annotated maps sharing personal opinions, experiences and emotions by ordinary users. However, existing tools seem to have diverged into two extremes. On one hand, there are programming-based map visualization scripts, which offer high flexibility and customizability, but also come with a steep learning curve. On the other hand, there are clip-art-style tools that allow users to place annotation elements on maps by simple interactions, reducing operational difficulty but often resulting in inefficiency, ambiguity and poor reusability. To address this, we propose GoMA, a structured grammar for expressing map annotation semantics. By decomposing the semantic space of annotations, GoMA structures the information to be annotated and further links it with geographic data and visual encoding strategies to generate annotated maps. Based on GoMA, we provide a framework for constructing annotated maps and demonstrate its advantages, including modularity, ease of extension and reusability, through case studies.

Index Terms—Map Annotations, Declarative Grammar, Semantic Space, Spatial Visualization.

1 Introduction

Maps have consistently been among the most effective means of conveying spatial information, widely used in various scenarios. By building upon basic cartographic maps——altering visual styles and adding supplementary annotation elements—people can create annotated maps or map visualizations that enhance the map's ability to encode custom data.

However, current tools for constructing annotated maps or map visualizations have diverged into two extremes. On one hand, tools typified by online clip-art-style annotation enable users to easily add visual elements onto maps through simple interactions, such as Social Explorer[1], Scribble Maps[2] and MapMaker[3]. While these tools lower the entry barrier for creating annotated maps, it weakens the semantic linkage between the annotations and the underlying geographic entities. Typically, the spatial relationships are loosely determined through user-specified positions, resulting in fragile semantic associations. This fragility imposes severe limitations on the exploration, browsing, dissemination, and reuse of annotated maps. For instance, zooming operations can introduce ambiguity during exploration and browsing, while difficulties in changing base maps, reusing similar annotation semantics, or modifying styles can impede reuse. On the other hand, more professional map visualizations often require programming proficiency [3,13]. Users must select appropriate visual encoding strategies based on data types and semantics and then implement these strategies through custom coding. While scripts created in this manner efficiently automate map visualizations and support rapid data replacement and style modifications, the programming requirement poses a barrier. Furthermore, significant changes in annotation elements resulted from changes in data type or semantics necessitate reprogramming, leading to considerable expense of time and effort.

Some tools have attempted to bridge these two extremes by enabling users to convert imported data into map visualizations through intuitive interactions [15,23,27]. However, existing tools offer limited map visualizations, typically restricted to choropleth or bubble maps, which are not enough to satisfy the users' needs. Map annotation elements have a broad design space to meet the rich semantic representational needs of users when constructing annotated maps. However, the semantic space of map annotations received insufficient attention. Annotation semantics form a crucial bridge, linking geographical entities with visual encoding strategies, thus directly impacting the expressiveness and usability of annotated maps. Clip-art-style methods tend to fragment annotation semantics, treating annotations as decorative elements positioned manually, with negligible integration with geographic context or structured semantic intent. Conversely, scripted or template-driven approaches predetermine annotation semantics, geographic associations, and visual encodings, greatly constraining user creativity and flexibility.

To address these issues, we categorize and summarize annotation semantics at an appropriate abstraction level, and introduce Grammar of Map Annotations (GoMA), a declarative grammar designed to construct annotated maps. GoMA supports structured and modular description of map annotation information by decomposing the material for constructing annotated maps into three parts: geographic data, visual element constructor functions and annotation scripts. GoMA allows flexible replacement of data, annotation semantics, or visual encoding strategies, thus ensuring the adaptability and reusability of annotated maps. For professional users, GoMA leaves room for customization, enabling creative visual encoding strategies through programming.

• *Hanning Shao, Haoyu Xu and Xiaoru Yuan are with National Key Laboratory of General Artificial intelligence and School of Intelligence Science and Technology, Peking University. E-mail: {hanning.shao, hanyu.xu, xiaoru.yuan}@pku.edu.cn. Xiaoru Yuan is also with National Engineering Laboratory for Big Data Analysis and Application, Peking University and PKU-WUHAN Institute for Artificial Intelligence, China. Xiaoru Yuan is the corresponding author.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

[1]https://www.socialexplorer.com/home/

[2]https://www.scribblemaps.com

[3]https://www.nationalgeographic.org/society/education-resources/mapmaker-launch-guide/

Simultaneously, tools derived from GoMA can offer non-expert users an extensive library of predefined visual encoding strategies, allowing the construction of complex annotated maps through straightforward, interactive semantic annotation operations.

2 Related Work

To situate our work, we review related research in two directions: annotated maps and visualization grammars. We identify a gap at the intersection——existing methods often lack structured semantic representations for annotations. To address this, we propose GoMA, a declarative grammar designed to construct annotated maps.

2.1 Map Visualizations and Annotated Maps

As one of the most effective methods to express spatial information, maps have been used ever since antiquity. Among all types of maps, annotated maps, which are maps with various annotations on them, are frequently used by individuals to convey to their audiences the geographic knowledge involved in other materials, including experiences, stories, analysis insights and so on. Those annotated maps emphasize special places, areas, and the stories on them, which would never been covered in traditional cartographic maps.

Researchers have already noticed the trend of combining maps with other materials for better expression and communication. Andrienko et al. [1] review the analysis tasks and methods towards the spatio-temporal data. The combination of modern computer techniques and conventional cartography brings new potentials and challenges for researchers and mapmakers. Simone et al. [25] compare the communication power between textual materials and annotated maps. They find that the maps provide users with a deeper understanding and a better ability to recall the represented phenomenon. Through annotations, mapmakers recreate conventional maps to encode more information in them. Maps are provided to audiences together with other materials like text, and those annotated maps thus become effective assistance for readers to understand the insights from authors.

Researchers from various fields have made attempts with annotated maps. Pearce [13] applies annotated maps in narratives to emphasize the places involved in trade voyageurs's trajectories. Shin et al. [23] developed a web-based authoring tool prototype for studying the creation of AR stories for outdoor cultural heritage sites. And other researchers combine annotated maps with news articles and multimedia. Stephens et al. [27] mark crowdsourced video-recorded stories about sea level rise on interactive maps, while Lindgren et al. [10] apply interactive maps to narrate community news for the public. Vujakovic et al. [29] also explore the role of news maps in geopolitical discourse.

With the rapid development of GIS techniques [9, 28], it becomes much easier for individuals to create their own annotated maps based on digital maps. Thus, more and more annotated maps are applied to help with communication and narration. Considering this phenomenon, researchers also explore the techniques and strategies to guide the construction and usage of annotated maps. Caquard et al. [4] discuss the relation between maps and narratives from the aspect of cartography. Roth et al. [17, 19] propose methods for constructing interactive maps. They explore the usability of those map interfaces and provide a review of those interactive cartographic and geo-visualization maps. Griffin et al. [8] introduce the design transferability and context of maps, which will guide the design of those maps for different situations.

Other works explore the tasks using annotated maps in their research fields. Song et al. [26] and Roth et al. [18] classify the themes, elements, genres and visual tropes applied in narrative maps and conduct user studies on individual audiences to study the influences. In earth sciences, Phillips et al. [14] conclude the eight basic plots that are required to be expressed. Fish et al. [6, 7] collect a dataset of annotated maps for climate change communication. They interview the mapmakers for their strategies for conveying data insight to the audiences. Sieber et al. [24] study the potential of public participation in the case of building geospatial maps. Antoniou et al. [2, 3] propose a web-based map case on a volcanic peninsula, introducing their research findings in the area. Sun et al. [5] show a partial case of a water resource survey. They also adopt techniques of spatial mapping in their expression and find out that their method can enrich the expressive forms by combining abundant information, which may provide references for future design.

The barriers to creating annotated maps have been reduced with the development of computer sciences and geography techniques. More and more people begin to add annotations on maps to convey their insights, stories and so on information to their audiences. Nearly all kinds of spatial-related information can be combined with annotated maps to improve the efficiency of expression and communication. Most of these works focus on the theories of how to use annotated maps, especially applying those annotated maps to storytelling tasks. However, the analysis of the map annotation design space still stays at a primary stage, where researchers only propose classification theories borrowed from other annotation designs, such as chart annotations. Map annotations should be strictly linked with geographic information and the design space of those map annotations requires systematical reviews and summaries. In this work, we categorize and summarize annotation semantics at an appropriate abstraction level and introduce GoMA, which decomposes annotation semantics and links them systematically to geographic entities and visual encodings.

2.2 Grammars for Visualizations

Visualization grammars provide a formal way to describe data graphics by breaking them into fundamental components. Visualization grammars are appealing, for they are abstract enough to make the specifications concise and easy to modify, but also flexible enough to produce many custom visualization designs [16]. A seminal example is Grammar of Graphics [31], which defines a system to concisely specify the components of a graphic, including data, aesthetics, statistics, geometry, scale, coordinate, and facet. Hadley Wickham's ggplot2 [30], a widely used R package, is a direct implementation of the Grammar of Graphics, using a layered grammar of graphics approach. Vega [20] is another general-purpose visualization grammar for interactive graphics on the web. Vega specifications are JSON documents that declaratively define a visualization's structure. Built on top of Vega is Vega-Lite [21], a higher-level grammar that simplifies common tasks. Vega-Lite is a high-level grammar of interactive graphics with a concise JSON syntax for rapidly creating typical charts.

Beyond general tools, many visualization grammars target specific domains or chart types to address specialized requirements. The OpenGIS Styled Layer Descriptor (SLD) Profile [11] of the OpenGIS Web Map Service (WMS) Interface Standard defines an encoding that extends the WMS standard to allow user-defined symbolization and coloring of geographic feature and coverage data. CartoCSS [12] is a style encoding which allows a high-level customization for the cartographic aspects of a map. It offers a compact and familiar syntax for cartographers to write styling rules, which can then be compiled to lower-level specifications. Florence [15] is built with existing open web standards

(HTML, CSS, JavaScript) and the JavaScript framework Svelte. Florence introduces custom Svelte components that correspond to marks and layers in a map, allowing developers to declaratively compose map visuals in code. Bertin.js[4] is a JavaScript library for visualizing geospatial data and making thematic maps for the web. Bertin.js makes it easy to produce common map types by providing high-level functions. Existing map visualization grammars and systems demonstrate capabilities in styling, layering, and rendering geospatial data with rich visual elements. Standards like SLD and tools like CartoCSS enable declarative control over map appearance, while frameworks like Bertin.js and Florence lower the barrier for thematic mapping and interactive design through domain-specific abstractions. However, these approaches primarily focus on visual styling and lack structures expressing the underlying semantics of annotations, which limits reusability, interactivity, and higher-level reasoning. To address this gap, our work introduces GoMA, a grammar of semantic annotations on maps, which enables structured, extensible, and semantically meaningful annotated map construction.

3 Methodology

In this section, we first outline the fundamental workflow for constructing annotated maps and provide a detailed explanation of the relevant concepts. The annotated maps examined in this study are based on cartographic maps. This requires that representation of geographic entities should retain their original spatial coordinate information, utilizing points, lines and polygons. The styling of map elements can be customized and additional visual elements can be added to enhance representational clarity. As a result, specialized forms of geographic representation, such as cartograms, are excluded from the scope of this work.

3.1 Materials

As with the construction of most data visualizations, creating an annotated map involves three essential components: data, visual encoding strategies, and the visual element constructors. By mapping each data dimension to suitable visual channels based on the chosen encoding strategy, corresponding visual elements are generated and integrated to form the final representation. Each of these components will be discussed in detail below.

```
type BaseGeoEntity  = Point | Line | Polygon | MultiPoint |
                      MultiLine | MultiPolygon
type GroupGeoEntity = GeoEntityList | GeoEntityCollection
type GeoEntity (GE) = BaseGeoEntity | GroupGeoEntity

func Annotation ( ... GE[], Data?) ⇒ VisualElements (VEs)     (B)

func AnnoState (GE, Data?) ⇒ VEs                              (S₁)
func AnnoState (BaseGeoEntity, Data?) ⇒ VEs                   (S₂)
func AnnoState (GroupGeoEntity, Data?) ⇒ VEs                  (S₃)
func AnnoState (GroupGeoEntity,
                (List<Data> | Collection<Data>)?) ⇒ VEs        (S₄)

func AnnoRelation ( ... GE[], Data?) ⇒ VEs                    (R₁)
func AnnoRelation (GE, GE, Data?) ⇒ VEs                       (R₂)
func AnnoRelation (GE, GroupGeoEntity, Data?) ⇒ VEs           (R₃)
func AnnoRelation (GE, GroupGeoEntity,
                   (List<Data> | Collection<Data>)?) ⇒ VEs     (R₄)

func Annotation ( ... GE[], Data?, Configuration?) ⇒ VEs      (E)
```

Fig. 1: Type definitions for geographic entities and declarations of visual element constructor functions are provided. Function overloading is employed to define these constructors. *GE* refers to *GeoEntity* and *VEs* denotes *VisualElements*. A question mark ('?') indicates optional parameters.

[4]https://github.com/riatelab/bertin

Geographic Data

The geographic data used in constructing annotated maps exhibits distinct characteristics: all data pertains to specific geographic entities, describing their attributes and interrelationships. This characteristics allow geographic entities to serve as a natural index for organizing geographic data. Based on the type of geographic entities, the data can be broadly categorized into three fundamental types: points, lines and polygons. To support simple combinations, additional classes such as multi-points, multi-lines and multi-polygons are included. These classifications are consistent with widely adopted geographic data formats, such as GeoJSON[5] and KML[6]. In addition to these basic categories, two extended types of geographic entity groups are introduced: ordered geographic entity lists and unordered entity collections. These serve as extensions to the foundational classifications. Fig. 1 presents the type definitions of *GeoEntity*.

With respect to geographic entities, geographic data encompasses three types of information: geometric coordinates, entity attributes and inter-entity relations. Coordinates and attributes are associated with individual geographic entities, whereas relations pertain to entity groups.

- **Geometric Coordinates** describe the spatial properties of basic geographic entities, capturing geometric details such as location and shape. These data are fundamental for rendering the underlying cartographic map.
- **Properties** encompass the attributes of geographic entities, such as name, area, and population. These non-geometric data can be ordinal, quantitative, or categorical, providing substantial flexibility in terms of data type.
- **Relation Properties** contain relational information among a group of geographic entities, functioning similarly to attributes but applying to interconnected entities rather than to individual, independent ones.

Comparing with the most commonly used geographic data format, GeoJSON, the aforementioned definitions provide enhanced support for richer relational information. Specifically, in GeoJSON, a *feature* defines an individual geographic entity, where the *geometry* specifies the entity's type and corresponding coordinate information, and the *properties* contain its attribute data. These components correspond directly to the first two categories of data described above. However, GeoJSON has limited capacity to represent relational information among geographic entities, primarily relying on hierarchical structures to express subordinate relationships. More complex multi-entity data, such as origin-destination (OD) data or multi-point trajectory data, cannot be adequately represented. These data correspond to the third category, which can be effectively described by the definition of entity groups. For instance, trajectory information can be structured as an ordered entity list, with *relation properties* specifying attributes such as speed or duration. Similarly, unordered entity collections can represent spatial relationships such as containment or membership among geographic entities. This extension significantly enhances the expressive capacity of geographic data beyond that of GeoJSON.

Visual Element Constructors

In the construction of annotated maps, data is transformed into various visual elements that are subsequently combined to produce the final representation. The tools employed to generate these visual elements are termed *Visual Element Constructors*. Functionally, these constructors accept geographic data as input

[5]https://geojson.org/

[6]https://developers.google.com/kml

and generate corresponding visual elements based on that data, as defined in Fig. 1 (B). The input geographic data encompasses coordinate information, attributes and relational data. Within the constructor, data are mapped to distinct visual channels and converted into visual elements. By appropriately integrating these constructors, all data intended for map annotations can be sequentially transformed into visual elements, thereby assembling the annotated maps.

Visual element constructors constitute essential components in building annotated maps, enhancing their extensibility and reusability. However, these constructors have received insufficient attention in existing tools and workflows. In interactive placement-based construction tools, user interactions implicitly perform the functions of these constructors by selecting appropriate visual elements and adjusting their positions and styles. Conversely, in map visualization scripting, all constructor functionalities are typically combined without modular separation into reusable components. For example, when creating a choropleth map that encodes population density through regional color gradients, the process of converting a regional geographic entity's coordinates into a polygonal visual element and coloring it based on the density value could be modularized as an independent, reusable constructor.

Extracting and modularizing these reusable constructors can substantially improve the efficiency of annotated map construction. This approach increases the extent of automated processing while minimizing redundant coding and repetitive interactive tasks. Furthermore, it enables easier modification and reuse of annotated map outputs.

Annotation Scripts

The visual encoding strategy defines how data are transformed into visual elements. We refer to the formal description of this strategy as *Annotation Scripts*. Specifically, annotation scripts comprise a series of operations for the visual encoding of data. These scripts establish the linkage between geographic data and visual element constructors. They select relevant subsets of geographic data and input them into the corresponding constructors, thereby completing the transformation from data to visual elements. In other words, annotation scripts specify how visual element constructors should be executed.

The annotation script itself need not address specific data details or the internal implementations of visual element constructors; rather, it focuses on decision-making and semantic alignment. An annotation script comprises a series of annotation operations, each involving two key components: the relevant data and the applied constructor. Each operation embodies operational semantics, specifically transforming designated *data* into visual elements via the encoding method provided by the assigned *constructor*.

The data includes indice of geographic entities and the keys pointing to corresponding coordinates, attributes or relational data of these entities. The constructor part declares the function index to be executed, as well as configure parameter specifications, which supports additional arguments required by the function beyond geographic data. These configure parameters typically serve as convenient means to externally influence internal encoding behaviors. For instance, as shown in Fig. 2 (C_1), when rendering glyphs for cities, the construct function may accept a *showLabel* parameter which controls whether the text labels should be rendered to accompany the city glyphs.

3.2 Annotation Semantic Space

The design space for map annotation is exceptionally rich, primarily due to the endless possibilities of visual element design.

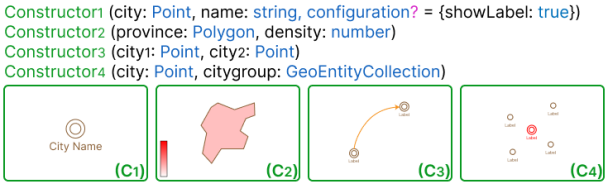

Fig. 2: Four examples of constructor functions, including their function declarations, parameter types, and output visual effects.

However, from the perspective of annotation semantics, it is limited. Specifically, annotation semantics address the following questions: in an annotated map, what geographic data is encoded into visual elements and through what means? This corresponds to the semantics of annotation scripts. By incorporating semantic information carried by the data, we can further refine the semantic space of map annotations.

The semantics of visual element constructors are reflected in two aspects. The inputs of these functions primarily consist of geographic data, which has already been considered in the data part. The other aspect involves the actual construction of visual elements, which pertains to visual encoding strategies and corresponds to the design space of visual elements. We just retain it with the constructor as an inherent feature of the function. Based on this, the following discussion focuses on analyzing the semantic space from the dimensions of data and annotation operations.

As mentioned, geographic data can be categorized into three types: entities' coordinates, attributes, and relations. Considering that annotations are all based on cartographic maps, the entity coordinates, which include the specific location and shape of geographic entities, are already reflected in the positional spatial semantics conveyed by the base map. Therefore, we exclude it from the core annotation semantics and focus primarily on the entity attributes and relations. Using annotation scripts to label this data thus involves two types of semantics: annotating entity states and annotating entity relationships.

Annotate States

The annotation of entity states is the operation that encodes the attribute information of geographic entities and transforms them into visual elements, as shown in Fig. 1 (S_1) extended from (B). Strictly speaking, the geographic entities in this context refer to individual entities, which can only belong to fundamental types. The visual element constructors take entities and their attributes as input and output corresponding visual elements, as shown in Fig. 1 (S_2). Here the geographic entity parameters typically include the entity's index and coordinate information, while the state parameter corresponds to the attribute dimension to be encoded.

The annotation semantics vary depending on the category of the geographic entity and the attribute dimension being processed. For example, Fig. 2 (C_1) represents annotating the *name* attribute of entity $City_1$ *(Point)*, while Fig. 2 (C_2) denotes annotating the population *density* attribute of entity *Province (Polygon)*.

The category of geographic entities can further be extended to a group of entities. This involves two distinct scenarios. The first treats the group of entities as a unified whole, expressing the attribute information of the collective, as shown in Fig. 1 (S_3). In some cases, this entity group can be decomposed to individual entities, meaning all entities in the group share the same value on that attribute dimension. For example, if a group of cities are all classified as first-tier cities, this can be interpreted either as the entire group being a first-tier city group or as each individual city

entity having the same classification level of *first-tier*. In other cases, such decomposition may not be applicable. For instance, if the boundary between two countries is represented by a series of connected *landmarks (Point)*, and the annotation emphasizes the line formed by these landmarks. The semantics cannot be simplified to merely stating that each individual landmark lies on the boundary.

When the geographic entity group exhibits varying values in the specified attribute dimension, the annotation operation can be interpreted as batch annotation of each individual entity within the group. In this scenario, the state parameter passed to the constructor should maintain the same indexing structure as the entity group, as illustrated in Fig. 1 (S_4). An ordered list of geographic entities should correspond to an equally ordered list of state parameters, while a collection entity group should align with collection state parameters. For example, Fig. 2 (C_1) can also be used to demonstrate annotating the *name* attribute for each entity $City_i$ *(Point)* within a city entity group with this aggregated method as Fig. 5 (*RenderCity*). This kind of expression serves as an aggregated simplification method for large-scale, similar annotation operations.

Annotate Relations

The annotation of relations is the operation that encodes relation information among geographic entities and transforms it into visual elements, as shown in Fig. 1 (R_1) extended from (B). The number of specific geographic entities involved in relation information is indeterminate. The following discussion uses the relation between two entities as the primary example, while cases involving more entities can be straightforwardly extended from this basis. Thus, the visual element constructors for relation annotations can be expressed as shown in Fig. 1 (R_2), taking two entity parameters, $entity_1$ *and* $entity_2$, and a relation data parameter as input, and outputting visual elements.

When both entity parameters belong to fundamental categories, the semantics are relatively straightforward, representing the annotation of relation between these two entities. For example, Fig. 2 (C_3) illustrates annotating movement trajectory information from $city_1$ *(Point)* to $city_2$ *(Point)*.

When one entity is extended to an entity group, either an ordered list or unordered set, the semantic interpretation becomes correspondingly more complex. Similar to state annotation, it is necessary to classify whether the entity group should be treated as a unified whole. Given the symmetric nature of the two entity parameters, the following discussion will use the case where $entity_2$ is an entity group as an example.

First, when the group is treated as a whole, the annotation semantics remain consistent with relation annotation between basic entities, while the corresponding constructor expression is shown in Fig. 1 (R_3). For instance, Fig. 2 (C_4) demonstrates annotating that $city$ is the most developed city within the group. In the second scenario, the entity group needs to be decomposed. This annotation operation can be interpreted as separately annotating the relation information between $entity_1$ and each individual entity within the entity group $entity_2$, serving as an aggregated simplification for numerous similar annotation operations. Here, we further consider whether the relation information is consistent across the group. That is, whether each entity in the entity group $entity_2$ shares the same value in the annotated relation dimension with $entity_1$. When it is identical, the relation parameter can be passed as a single instance, corresponding to the format shown in Fig. 1 (R_3). While when it varies across the dimension, the relation parameters must maintain indexing consistency with the geographic entity group, as demonstrated in Fig. 1 (R_4).

Extending this further, when both entity parameters are entity groups, the semantic interpretation of the annotation can be analyzed by first treating one of them as a unified whole to examine its annotation semantics with the other one. Then further decomposition of entity group can be done towards this parameter.

Conclusion

We have established the semantic space for annotation operations in map annotation construction. First, based on the category of geographic data being annotated, this space is decomposed into two major classes: annotating geographic entity attributes and annotating entity relations. Subsequently, each category is further analyzed in detail according to the specific types of entities and information involved in the annotation operation. This process also demonstrates the advantages of introducing the concept of geographic entity groups. First, groups expand the semantic coverage of annotations, enabling multiple geographic entities to be treated as a unified whole and assigned attribute information or relation information with other entities. Besides, groups facilitate the aggregation and simplification of similar repeated annotation operations, thereby improving the expressive efficiency of annotation scripts and reducing the cost of construction and modification.

4 Generate Annotated Maps

This section will present a formal grammar for constructing annotated maps based on the analysis and definitions mentioned above. Building upon this grammar, we further propose a workflow framework for constructing annotated maps.

```
GoMA: {
  GeoData: {
    [key: BaseGeoEntity.ID]: {Coordinates, Attributes},
    [key: GroupGeoEntity.ID]: {Contents, Relations},
  },
  Libs: {
    [key: Constructor.ID]: {
      FunctionPointer,
      ParameterDeclaration,
      ConfigurationDeclaration,
    }
  },
  Scripts: Array<{
    Data: {
      Entity: GE.ID | GE.Contents,
      Keys: (Attribute.Key | Relation.Key)[],
    },
    Func: {
      ID: Constructor.ID, Configuration?: any,
    }
  }>
}
```

Fig. 3: The structure of GoMA, including data part (*GeoData*), function declarations part (*Libs*) and annotation operations part (*Scripts*). '?' means the attribute that is optional.

4.1 Grammar of Map Annotations

This grammar for constructing annotated maps comprises three components, geographic data, function declarations for visual element constructors, and annotation scripts, as shown in Fig. 3. Among these, the data and function declarations remain relatively independent, while the annotation scripts will reference and correlate the other two parts to describe the operational workflow for creating annotated maps.

GeoData employs geographic entities as indices and categorizes them into basic geographic entity data and geographic entity group data. Basic geographic entities are indexed by individual

geographic entities, containing both coordinate information and attribute information. The coordinate information records the specific category of the entity along with the corresponding geometric coordinate data. The attribute information may include multiple attributes, distinguished by attribute names. There are no further constraints imposed on the specific types of attribute values. Theoretically all types of data can be assigned as the value for a geographic entity's attribute, including but not limited to string, number, boolean, vector, matrix, etc., which can be freely defined according to practical requirements.

Geographic entity group data entries form their indices with the entities in the group. In practice, indexing can be achieved either by combining the indices of the constituent entities or by assigning an independent unique identifier. The data includes the group contents as well as corresponding relation information. The group content will be either a list or a collection, depending on the organizational structure of the entity group, with only the indices of the constituent entities recorded. Relational information is analogous to the attribute information of basic entities, except that it applies to the entity group as a whole. Different relations are distinguished by their names. Similarly, the actual value types of relation information can be flexibly specified according to practical requirements.

Basic geographic entity data are self-contained, while entity group data reference other data items in their content field, including both basic entities and other entity groups. All geographic entity data entries possess unique identifiers, and form a directed acyclic dependency graph.

Libs is a function declaration, containing an index of all available visual element constructors. Each of the functions has a unique identifier and should belongs to the categories defined previously in the annotation semantic space. The number and types of input parameters should meet the corresponding constraints, and all functions output visual elements.

In GoMA, **Libs** field works as function declaration header files. It records the specific input requirements of each function and provides pointers to the executable functions themselves. The internal implementation details of these functions are not included. A configuration declaration is also included to provide information on the supported configurations.

Script is an array in which each entry records an annotation operation, namely passing specified geographic data to the assigned constructor function to generate corresponding visual elements. The operations should be executed sequentially, with the resulting visual elements naturally overlaid to form the annotated map. Two components are included in each operation entry, the data specification and function specification.

In the *Data* field, geographic entity indices and attributes or relation information keys are provided. Specifically, the *Entity* part contains the indices of the geographic entity or group to be processed. Depending on annotation requirements, this may consist of a base geographic entity, a list or a collection of entities. Temporary lists and collections can be formed here to construct a temporary entity group in cases when the corresponding annotation operations are semantically aggregated (*GE.Contents*). And these temporary entity groups will be used only for the current annotation operation. In contrast, passing the indices of an entity group as parameters means that the entity group should be regarded as a whole.

The data keys inputed represent the state or relation data involved in the annotation operation. The *Keys* field records the data names of the attribute or relation data being referenced. When a temporary entity group is constructed in the *Entity* field, the data keys recorded in this field should be interpreted as directly referencing each individual entity within the group.

In the *Func* field, the index of the constructor and configuration are included. The constructor index directly references the corresponding entry in the function library and thus points to the executable function. To further enhance the flexibility and extensibility of these constructors, in addition to the geographic entity parameters and the state or relation parameters specified in the annotation semantics, these construction functions are allowed to provide a configuration interface. The configuration can influence the function's behavior differently from the input geographic data. They are not part of the data but rather relate to internal visual encoding details within the functions. Configuration is optional and if omitted, the function should provide default values. When executing the annotation script, the configuration is passed to the function along with other geographic data, as shown in Fig. 1 (*E*).

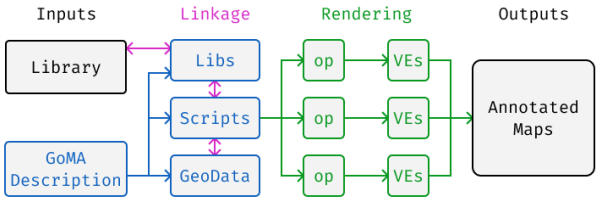

Fig. 4: The workflow of generating annotated maps. It takes constructor function library and GoMA descriptions as input. Purple arrows represent linkages, while the green ones indicate execution of constructor functions.

4.2 Workflow

Based on the grammar, we propose a feasible workflow for constructing annotated maps, as illustrated in Fig. 4. The input consists of two components, a structured description following the grammar and a visual element constructor function library. The output will be the complete annotated maps. The core process is to parse the annotation scripts in the description, linking the geographic data with the constructing functions. After the linkage, the step-by-step execution of those functions will build visual elements according to the script and ultimately accomplish the annotated map.

The first step is to link the data and function with the scripts in the description. Based on the geographic entity indices and parameter keys contained in the *Entity* fields, corresponding geographic data entries can be retrieved from the *GeoData* field. The acquired data includes geometric coordinates for each entity, which is often default, along with the state or relation information specified by data keys. Besides, the constructor indices specified in the *Func* fields are used to retrieve corresponding executable functions from the function library. After the linkage, geographic data, including entity coordinates, state or relation data, along with the configuration can be passed to the visual element constructor as input arguments. The constructor then generates the visual elements accordingly. By sequentially executing all function calls as defined in the script, all annotation elements are obtained and combined to produce the annotated map.

Fig. 5 shows a case of annotation description for constructing a trajectory annotation map, where the *Scripts* field contains three operations. The first step involves rendering the base map of provincial regions. The input geographic data consists of an entity group comprising all provincial entities, requiring only coordinate information without additional state data. The specified constructor function is a polygon collection rendering function.

The second step involves plotting key city points on the map. The input data consists of a temporarily constructed entity group,

namely a collection of key cities. In addition to city points' coordinates, city names are required as state parameters. Since the geographic entity parameters form a temporary entity group, the state parameter specification in the script should be directly applied to each entity within the group. Consequently, the linked parameters comprise collections of name attributes of each key city. The constructor function assigned for this operation is designed to plot city glyphs for point entities and append text labels beside them. The whole operation should be regarded as an aggregation of plotting all city points on the map.

The final step involves trajectory rendering. The trajectory route is defined in the data through an ordered entity list, which is passed to the function as geographic entity data. The constructor processes this entity list by rendering arrow elements between adjacent pairs of points to visualize the path. This function supports arrow color specification through configuration. Thus, the script designates the arrow color as blue in the configuration field. Since this color assignment encodes no geographic information and merely adjusts visual styling, it is implemented as part of the configuration rather than data inputs.

The visual elements constructed by the three-step script are sequentially overlaid to produce the final trajectory map.

```
Scripts: [
  {
    Data: {
      Entity: AllProvinces, Keys: []
    },
    Func: {ID: 'RenderProvinces'},
  },          RenderProvinces (entity: GeoEntityCollection)
  {
    Data: {
      Entity: {City1, City2, ...}, Keys: ['name']
    },
    Func: {ID: 'RenderCity'},
  },              RenderCity (entity: Point, cityname: Data)
  {
    Data: {
      Entity: Trajectory, Keys: []
    },
    Func: {
      ID: 'RenderTrajectory',
      Configuration: {ArrowFill: 'blue'},
    },
  },
]   RenderTrajectory (entity: GeoEntityList, configuration)
```

Fig. 5: An example of *Scripts* which contains three annotation operations and is used to build a trajectory map. The function declarations related to the *Func* fields are also provided, marked by stroked borders.

5 Usage Scenario

In this section, we validate the feasibility of GoMA as well as the corresponding workflow through a case study involving automated text-based map annotation generation to assist text-map cross-referenced reading. We demonstrate the advantages of GoMA in structured and modular annotation construction, improving the editing, modification, and reuse of annotated maps.

The data used in this case study are sourced from a historical geography research paper. Shen's work [22] proposes new insights into the boundary division between Shunning Prefecture and Yongchang Prefecture in southwestern China during the Qing Dynasty by incorporating additional evidence from local chronicles.

It is noted that in the textual research of historical geographical literature, the content typically focuses on two types of evidences. The first type emphasizes the status of a single or a group of geographical entities, particularly concerning their jurisdictional affiliations. The second type highlights the relations or events between entities, such as the adjacency of areas, directional and distance relations between cities, or shared historical events.

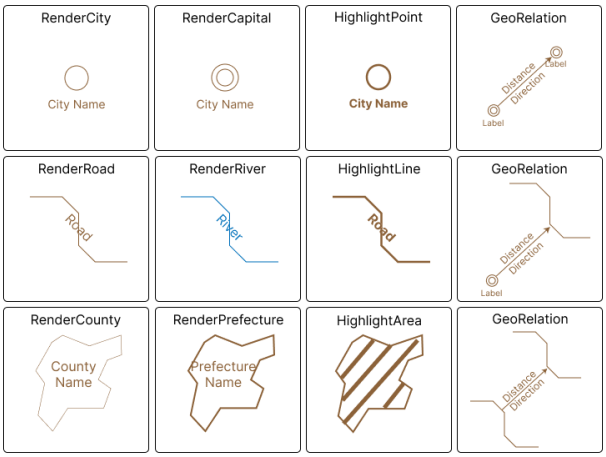

Fig. 6: Pre-defined visual element constructors used in the case. Six *RenderXX* constructors are used to generate the base map entities. The *HighlightXX* ones are for state annotation, while the *GeoRelation* ones are applied to emphasize the geographic relations between various types of entities.

By automatically constructing annotated maps, corresponding elements can be generated based on the textual content to facilitate text-map cross-referenced reading. These two types of textual content can be represented by predefined visual elements, while large language models (LLMs) can be employed to extract and structure the information from unstructured text. GoMA description can be generated according to predefined rules by integrating these two parts, thereby fulfilling the requirement for automated annotated map generation.

Fig. 6 shows the predefined visual element constructors, which can be divided into three categories, generating annotations for the base map, individual entity's emphasis, and spatial relations between pairs of entities respectively. Considering reading assistance does not require detailed semantic annotations on the map, the focus remains on highlighting relevant entities. Based on the categories of geographical entities, including points, lines and polygons, distinct highlighting constructors are designed.

In terms of text information extraction and structuring, tailored prompts for LLMs are designed. The prompt templates, as shown in Fig. 7, include task descriptions, examples and the specific data to be processed. The output of LLMs consists of structured information that can be directly transformed to GoMA styles.

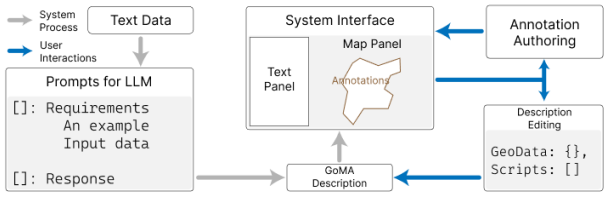

Fig. 7: The workflow of the system introduced. It leverages LLM to extract and structure the geographic information from text and generates annotated maps to help with reading. Users can also edit the GoMA description during this process.

We implemented an automated annotated maps generation system accordingly, with its workflow illustrated in the Fig. 7. The text and base map data (coordinate data in GeoJSON format) are input into the system. The textual content is segmented into individual sentences, and then be processed by LLMs with corresponding prompts generated using predefined templates. The LLMs return structured geographical information, which is then combined with the base map data to generate the *GeoData* field of GoMA. Meanwhile, the *Func* field is also predefined according to the constructors shown in Fig. 6.

Users can interact with the system by clicking on any sentence. The system retrieves the geographical information contained in the corresponding sentence, generates an annotation operation script accordingly, and renders it into an annotated map. When no sentence is selected, the annotation script only includes operations for generating annotations on the base map. Furthermore, the system provides an editing panel, allowing users to modify the map annotation scripts to address possible errors arising from the automated process. Fig. 8 demonstrates the effects for the base map and five individual sentences.

This case demonstrates the advantages of structuring and modularizing the annotated map construction process. First, modification of annotated content only requires replacing the geographical data and scripts in the description. In this example, text sentences yield distinct geographical information. All such data is recorded in the *GeoData*, and the system merely updates the *Scripts* when users switch focus between sentences. Second, the final presentation of the annotated map can be adjusted simply by updating the visual element constructors, since the visual styling remains independant from annotated content. By modifying the constructors' implementation or switch functions assigned by the scripts, the map's visual style is updated. Lastly, GoMA isolates the geometric coordinate data in the base map. Subsequent updates to the base map, such as refining regional boundary precision, will not disrupt references from other components.

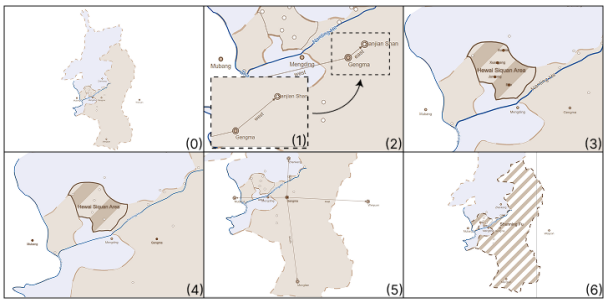

Fig. 8: Annotated maps generated by the system. Map_0 is the base map, with no sentence annotated. Map_2 to Map_6 demonstrate the annotation results of five different sentences. In Map_2, there is an error from the original text. Users can edit the GoMA description to correct it, as shown in Map_1.

6 Discussion

In this work, we propose GoMA, a grammar for constructing annotated maps. From semantics perspective, we summarize the semantic space of data and annotation operation within the workflow and decompose annotation materials into three parts, including geographic data, visual element constructors and annotation scripts. Thus, GoMA enables structured and modular descriptions of map annotations. The approach enhances both the efficiency of building and modifying annotated maps, while simultaneously improving their reusability.

Structured the materials of generating annotated maps. GoMA reorganizes geographic information data around entities and structures the semantic space of annotations. Beyond individual entities, we introduce the concept of geographic entity groups, significantly enriching data representation capabilities. Thus, compared to popular data formats like GeoJSON, GoMA can further encompass multi-entity relation information without requiring additional tables or other forms of data expansion. In terms of generating annotated maps, GoMA decomposes the visual encoding process into two parts: specific annotation scripts and reusable visual element constructors. This provides unprecedented flexibility for real-world scenarios, such as seamlessly replacing data, map styles, or visual encoding strategies. Different constructors can be substituted freely to achieve varied visual effects without altering the annotated data or scripts, provided the replacement ones maintain identical operational semantics. Conversely, the visual style may be retained while replacing the underlying information to be annotated, enabling rapid generation of thematic maps with consistent aesthetics.

Extensible visual element constructor function library. Given the relatively limited variety of data types and annotation operation semantics, a predefined external library of visual element constructors plays a critical role in ensuring the richness of map annotations. Under the GoMA framework, this function library can be continuously expanded and refined. Any newly appended function can seamlessly integrate into the workflow, provided it adheres to the specified annotation operation semantics and offers compatible input parameter interfaces. Right in this work, we demonstrate only a limited set of concrete constructor functions. Developing a more comprehensive and scalable function library requires sustained effort and represents a long-term endeavor, which would be our future direction.

Flexible implementation strategies. GoMA structures the materials used for generating annotated maps and provides a workflow based on it. However, the specific implementation of GoMA's compiler and constructor library can be flexibly adjusted according to the requirements of different usage scenarios. This paper primarily presents examples of 2D SVG-format maps, where the output visual elements of the constructors are correspondingly SVG elements. Depending on different usage scenarios, modifications can be made. For example, switching from SVG to canvas rendering can significantly improve rendering efficiency to accommodate larger-scale data annotation demands. If the constructor's output is changed to 3D grids and transfer functions, and the visual element composition process is modified to volume rendering, GoMA can also be used to construct 3D maps. Naturally, the constructor library should be updated accordingly with the workflow's implementation.

Diverse usage scenarios. GoMA can be applied to various scenarios for various targeted users, including the automatic map annotation generation demonstrated previously. For programmers, GoMA provides a map generation engine and an extensible, reusable visual element constructor library. They can control the generation by editing GoMA descriptive files or enrich the constructor library by programming to meet customized requirements. For other users, generating maps by modifying descriptive files significantly lowers the learning barrier. Developers can further create beginner-friendly UIs for GoMA, further reducing the difficulty of editing these descriptive files.

Acknowledgments

This work was supported by NSFC No. 62272012 and Wuhan East Lake High-Tech Development Zone National Comprehensive Experimental Base for Governance of Intelligent Society.

References

[1] N. Andrienko, G. Andrienko, and P. Gatalsky. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14(6):503–541, Dec. 2003. doi: 10.1016/S1045-926X(03)00046-6 2

[2] V. Antoniou, P. Nomikou, P. Bardouli, D. Lampridou, T. Ioannou, I. Kalisperakis, C. Stentoumis, M. Whitworth, M. Krokos, and L. Ragia. An Interactive Story Map for the Methana Volcanic Peninsula:. In *Proceedings of the 4th International Conference on Geographical Information Systems Theory, Applications and Management*, pp. 68–78. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, 2018. doi: 10.5220/0006702300680078 2

[3] V. Antoniou, L. Ragia, P. Nomikou, P. Bardouli, D. Lampridou, T. Ioannou, I. Kalisperakis, and C. Stentoumis. Creating a Story Map Using Geographic Information Systems to Explore Geomorphology and History of Methana Peninsula. *ISPRS International Journal of Geo-Information*, 7(12):484, Dec. 2018. doi: 10.3390/ijgi7120484 1, 2

[4] S. Caquard and W. Cartwright. Narrative Cartography: From Mapping Stories to the Narrative of Maps and Mapping. *The Cartographic Journal*, 51(2):101–106, May 2014. doi: 10.1179/0008704114Z.000000000130 2

[5] Feiran Sun, Xi Tang, Tianyu Ye, and Feng Zhu. Thematic atlas information expansion design: A storytelling concept under web environment. In *2015 23rd International Conference on Geoinformatics*, pp. 1–6. IEEE, Wuhan, June 2015. doi: 10.1109/GEOINFORMATICS.2015.7378701 2

[6] C. Fish. Storytelling for Making Cartographic Design Decisions for Climate Change Communication in the United States. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 55(2):69–84, June 2020. doi: 10.3138/cart-2019-0019 2

[7] C. S. Fish. Cartographic content analysis of compelling climate change communication. *Cartography and Geographic Information Science*, 47(6):492–507, Nov. 2020. doi: 10.1080/15230406.2020.1774421 2

[8] A. L. Griffin, T. White, C. Fish, B. Tomio, H. Huang, C. R. Sluter, J. V. M. Bravo, S. I. Fabrikant, S. Bleisch, M. Yamada, and P. Picanço. Designing across map use contexts: a research agenda. *International Journal of Cartography*, 3(sup1):90–114, Oct. 2017. doi: 10.1080/23729333.2017.1315988 2

[9] M. Haklay, A. Singleton, and C. Parker. Web Mapping 2.0: The Neogeography of the GeoWeb: Web Mapping 2.0 and neogeography. *Geography Compass*, 2(6):2011–2039, Nov. 2008. doi: 10.1111/j.1749-8198.2008.00167.x 2

[10] A. Lindgren and C. Wong. How is your Toronto neighbourhood portrayed in the news? Check it out using these interactive maps. Aug. 2015. 2

[11] M. Lupp. Styled layer descriptor profile of the web map service implementation specification [corrigendum]. version 1.1. 0 (revision 4). 2007. 2

[12] S. Ojagh, S. Saeedi, J. Badger, and S. Liang. Symbology encoding using cartocss to support complex symbols. In *Spatial Knowledge and Information Canada*, vol. 7, 2019. 2

[13] M. W. Pearce. Framing the Days: Place and Narrative in Cartography. *Cartography and Geographic Information Science*, 35(1):17–32, Jan. 2008. doi: 10.1559/152304008783475661 1, 2

[14] J. Phillips. Storytelling in Earth sciences: The eight basic plots. *Earth-Science Reviews*, 115(3):153–162, Nov. 2012. doi: 10.1016/j.earscirev.2012.09.005 2

[15] A. Poorthuis, L. van der Zee, G. Guo, J. H. Keong, and B. Dy. Florence: a web-based grammar of graphics for making maps and learning cartography. *Cartographic Perspectives*, (96):32–50, 2020. 1, 2

[16] X. Pu, M. Kay, S. M. Drucker, J. Heer, D. Moritz, and A. Satyanarayan. Special interest group on visualization grammars. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–3, 2021. 2

[17] R. E. Roth. An Empirically-Derived Taxonomy of Interaction Primitives for Interactive Cartography and Geovisualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2356–2365, Dec. 2013. doi: 10.1109/TVCG.2013.130 2

[18] R. E. Roth. Cartographic Design as Visual Storytelling: Synthesis and Review of Map-Based Narratives, Genres, and Tropes. *The Cartographic Journal*, 58(1):83–114, Jan. 2021. doi: 10.1080/00087041.2019.1633103 2

[19] R. E. Roth and M. Harrower. Addressing Map Interface Usability: Learning from the Lakeshore Nature Preserve Interactive Map. *Cartographic Perspectives*, (60):46–66, June 2008. doi: 10.14714/CP60.231 2

[20] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Computer graphics forum*, vol. 33, pp. 351–360. Wiley Online Library, 2014. 2

[21] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350, 2016. 2

[22] K. Shen. Historical atlas of china: Qing dynasty - maps of yunnan: Four new pieces of research on the boundary between shunning and yongchang prefecture, langqu, and nandian chieftain's territory. *Historical Geography Research*, 42(1):140–147, 2022. 7

[23] J.-E. Shin and W. Woo. How Space is Told: Linking Trajectory, Narrative, and Intent in Augmented Reality Storytelling for Cultural Heritage Sites. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–14. ACM, Hamburg Germany, Apr. 2023. doi: 10.1145/3544548.3581414 1, 2

[24] R. E. Sieber, P. J. Robinson, P. A. Johnson, and J. M. Corbett. Doing Public Participation on the Geospatial Web. *Annals of the American Association of Geographers*, 106(5):1030–1046, Sept. 2016. doi: 10.1080/24694452.2016.1191325 2

[25] F. D. Simone, R. Presta, and F. Protti. Evaluating Data Storytelling Strategies: A Case Study on Urban Changes. 2014. 2

[26] Z. Song, R. E. Roth, L. Houtman, T. Prestby, A. Iverson, and S. Gao. Visual Storytelling with Maps: An Empirical Study on Story Map Themes and Narrative Elements, Visual Storytelling Genres and Tropes, and Individual Audience Differences. *Cartographic Perspectives*, Aug. 2022. doi: 10.14714/CP100.1759 2

[27] S. H. Stephens and D. P. Richards. Story mapping and sea level rise: listening to global risks at street level. *Communication Design Quarterly*, 8(1):5–18, May 2020. doi: 10.1145/3375134.3375135 1, 2

[28] D. Sui and M. Goodchild. The convergence of GIS and social media: challenges for GIScience. *International Journal of Geographical Information Science*, 25(11):1737–1748, Nov. 2011. doi: 10.1080/13658816.2011.604636 2

[29] P. Vujaković. The State as a 'Power Container': The Role of News Media Cartography in Contemporary Geopolitical Discourse. *The Cartographic Journal*, 51(1):11–24, Feb. 2014. doi: 10.1179/1743277413Y.0000000043 2

[30] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Use R! Springer International Publishing, 2016. 2

[31] L. Wilkinson, D. Wills, D. Rope, A. Norton, and R. Dubbs. *The Grammar of Graphics*. Statistics and Computing. Springer New York, 2005. 2