A streamlined GIS interface for Citizen Science activities: QGIS Light

Serkan Girgin[1], Jay Gohil[1], Indupriya Mydur[1]

[1] Faculty ITC, University of Twente, 7522 NH, Enschede, The Netherlands – (s.girgin, j.h.gohil, i.mydur)@utwente.nl

Keywords: Citizen Science, Data Literacy, GIS, QGIS, User Experience, Usability.

Abstract

Citizen science has emerged as a powerful way to involve the public in scientific research, especially in domains like environmental sciences, where participants actively collect data in the field. However, citizen scientists can contribute beyond data collection by engaging in data analytics to generate meaningful insights in collaboration with researchers. To support this broader participation, user-friendly and accessible tools for data visualization and analysis are essential. This study addresses this need in geospatial tools by introducing QGIS Light, a simplified version of the most widely used free and open-source desktop GIS software, QGIS. Developed as a plugin to QGIS, QGIS Light offers a streamlined working environment by simplifying the user interface, removing non-essential and advanced features, and introducing additional tools by default to support basic needs, such as accessing base maps and creating charts. The paper begins with an analysis of the QGIS user interface with a focus on simplicity of use. It then outlines the specific actions required to enhance the user experience for non-technical users. The logic and technical implementation of the QGIS Light plugin are subsequently described in detail. Finally, additional user interface challenges in QGIS that affect overall usability are discussed. The findings highlight the value of critically evaluating existing interface elements and refining them into a more cohesive and standardized experience. QGIS Light is a first step in this direction to enhance usability of QGIS and may also guide similar simplification efforts in other GIS software, helping to lower their typically steep learning curves.

1. Introduction

Citizen science has become a powerful approach for engaging the public in scientific research, particularly in environmental monitoring (Conrad and Hilchey, 2011). Activities such as tracking air quality, mapping biodiversity, and assessing water quality benefit significantly from citizens going outdoors to collect data. In this regard, geospatial tools are essential for ensuring precise and efficient data collection (Arias de Reyna and Simoes, 2016). Beyond data collection, geospatial tools also enable effective data visualization and exploratory data analysis, allowing users to overlay different datasets, revealing spatial relationships and trends that may not be immediately apparent. This analytical capability empowers citizens to help generate meaningful insights and support evidence-based policies together with researchers and policymakers (Kocaman et al., 2022). Such a progression from data collector to data analyst fosters deeper engagement and motivation, further reinforcing the impact and sustainability of citizen science initiatives.

However, a key challenge in supporting these more advanced activities is making spatial analysis functions easily accessible to non-experts. Many GIS software offer powerful features that can support citizen science by allowing users to explore real-world datasets and analyse spatial relationships. But non-technical users with limited exposure to geospatial software often struggle with complex interfaces and the technical nature of these tools. This is especially the case for elders and young students, who may have limited data literacy or experience with data analysis methods (Haklay, 2013). Tasks like accessing and managing datasets, performing spatial analyses, and visualizing results typically require specialized training and guidance due to the steep learning curves of technical user interfaces and exposure to advanced concepts. Case studies have demonstrated that more accessible interfaces can enable previously underserved groups to participate more actively in citizen science activities (Bonney et al., 2014). Therefore, developing intuitive and simple interfaces for core GIS functions can lower barriers to spatial data exploration and increase participation from such groups in geo-citizen science initiatives.

Free and Open-Source Software (FOSS) plays a vital role in these activities by providing accessible spatial analysis tools to researchers, citizen scientists, NGOs, and budget-constrained agencies, helping them avoid vendor lock-in and licensing costs (Coetzee et al., 2020). QGIS is the leading FOSS desktop GIS application used for viewing, editing, and analysing geospatial data. It supports a wide range of vector, raster, and database formats, and is highly extensible through a robust plugin system (QGIS.org, 2025). It is widely adopted due to its rich feature set, active development community, and strong support for open standards. However, QGIS's interface is designed for users who need full control over GIS data manipulation and analysis. As a result, it includes dozens of menus and toolbars with many icons and dropdowns, multiple dockable panels, hundreds of processing tools, and a high number of customization options. A typical QGIS session can include layers panel, browser panel, attribute table, processing toolbox, map composer, and more, all of which can clutter the screen. While useful for advanced users, this sheer volume of components and tools with various configuration options adds complexity and increases the chance of getting "lost" in the interface for someone new to GIS.

For citizen science initiatives aimed at non-technical audiences across diverse demographics, particularly youth and seniors, a simplified, less overwhelming interface is essential, as recommended by established best practices (Skarlatidou et al., 2019). This is especially important considering that they need to understand at the same time core concepts, such as coordinate reference systems (CRSs), vector and raster data, or attribute tables. To reduce this barrier, we developed a QGIS plugin, QGIS Light, which simplifies its user interface and tailor it to the specific needs of non-technical users. We started by defining user stories that capture the user needs and translated them into technical design constraints. Next, we assessed the functionality and complexity of QGIS components and features against these requirements. Based on this assessment, we determined the essential steps to simplify and improve the user experience for the target audience. These steps were implemented through a user-friendly QGIS plugin that enables seamless switching to a simplified interface and perform basic GIS tasks with ease.

This paper presents our experience in designing a practical and intuitive user interface for QGIS that facilitates quick learning and user-friendly interaction. It begins with an overview of the QGIS user interface and its existing customization capabilities, outlining both their strengths and limitations. We introduce the user stories and corresponding technical design constraints that shaped our development approach, followed by a presentation of the actions taken to simplify the interface. We then provide a detailed explanation of the logic and technical implementation of the QGIS Light plugin. The paper concludes with a discussion of broader usability challenges in QGIS and outlines directions for future improvement.

2. Analysis of the QGIS User Interface

QGIS uses Qt, a cross-platform FOSS widget toolkit, as the framework for its graphical user interface (GUI) (The Qt Company, 2018). Qt provides ready-made widgets (e.g., buttons, sliders, menus, etc.) and tools to build a modern and complex GUI, and enables QGIS to run seamlessly on Windows, macOS, and Linux without rewriting the GUI for each system. It is also used by QGIS plugins, especially those written in Python, to build custom dialogs and forms, allowing developers to extend QGIS with sophisticated interfaces.

A typical Qt application has a menu bar with text menus at the top, a set of toolbars with icon-based buttons for quick actions, side panels that can be docked, moved or hidden, a main content area, and a status bar at the bottom showing real-time feedback. QGIS uses these components to provide a flexible user interface for working with spatial data (Figure 1). Main components are:

- *Menu bar* provides access to all functions grouped under menus like Project, Edit, View, Layer, Settings, etc.
- *Toolbars* offer quick access to common tools for navigation, editing, measuring, and managing layers.
- *Layers panel* displays a list of loaded layers (e.g., vector, raster, etc.) and allows users to manage their visibility, order, grouping, and styling.
- *Map canvas* reflects the active layers and map extent. It is the central area where spatial data is visualized and interacted with.
- *Browser panel* allows users to browse and load spatial datasets from local files, databases, and online services.
- *Status bar* shows coordinate information, map scale, rendering status, and access to CRS settings.
- *Processing toolbox* provides access to geoprocessing tools, including analysis, conversion, and data management algorithms.

Furthermore, QGIS includes various task-specific panels, such as those for layer styling, managing spatial bookmarks, performing undo/redo operations, and editing vertices. Some components allow multiple instances. For example, users can create additional map canvases linked to the main canvas to enable secondary views for navigation or comparison. Some advanced features have their own full-window interfaces, such as the *Print Layout* for producing high-quality cartographic outputs or the *Model Builder* for developing geospatial models interactively. This study, however, focuses on simplifying the main QGIS GUI and does not cover these specialized interfaces.

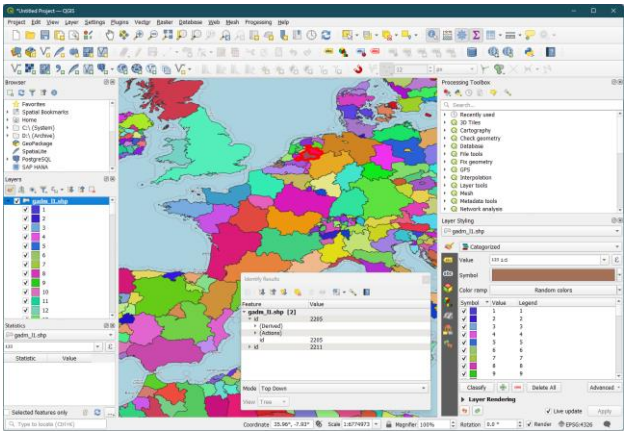

Figure 1. A typical QGIS session.

QGIS provides several GUI customization options to tailor the environment to specific workflows, preferences, and user needs.

Leveraging standard Qt capabilities, users can show or hide toolbars and panels, dock or undock them, and reposition these elements to create a personalized workspace layout. Through the *Options…* dialog found under the *Settings* menu, the overall appearance of the interface can be modified by selecting different visual styles, either native (e.g., Windows) or cross-platform (e.g., Fusion), as well as UI themes like *Night Mapping*. Users can also adjust font type and size, configure icon sizes, and customize various map canvas settings, colour schemes, and rendering options. Additionally, the GUI language can be set, which is especially useful in multilingual settings or training scenarios; changing the language also affects the display format for numbers, dates, and currencies.

Interface Customization… dialog, also located under *Settings* menu, allows users to enable or disable specific data sources, panels, menus, toolbars, status bar elements, and even individual widgets. Disabling an item removes its functionality across all related interfaces. For example, turning off a widget also disables its corresponding menu entry and toolbar button. Alongside layer details, map canvas states, and plugin statuses, layout and interface customizations are saved within QGIS project files. This ensures a consistent user experience across sessions when working on the same project. Moreover, QGIS remembers the last layout settings and automatically restores them in the following session. To support customized and consistent preferences across different projects, QGIS introduced *user profiles* in version 3.2. This feature enables users to create and manage multiple profiles, each with distinct settings, plugins, and configurations. Later releases, such as version 3.32, enhanced this functionality by adding a user profile selector, simplifying profile switching directly from the interface.

Despite offering extensive customization, QGIS has some notable limitations. A major gap is the lack of a built-in tool for creating custom toolbars to reorganize or group existing tools. This shortcoming is somewhat mitigated by the *Customize ToolBars* plugin (Raga, 2019), which permits users to build new toolbars by adding menu items, toolbar buttons, and processing algorithms. However, the plugin cannot modify existing toolbars or group buttons into dropdown menus, which are features that are crucial for decluttering the interface and making it more accessible, particularly for novice users.

In addition to the user interface, a crucial element influencing user experience and facilitating effective use of an application is the availability of comprehensive and easy-to-understand help resources and documentation. While extensive documentation exists for QGIS, it is often technical for younger or older learners who prefer visual, hands-on learning experiences. The integration of documentation within the application is also limited, as it is often shown in a separate web browser window outside of the application itself. Moreover, QGIS does not offer in-app tutorials or progressive onboarding. There is little real-time feedback or coaching to help users learn by doing.

The geospatial data analysis and processing features of QGIS are primarily accessed through the *Processing Toolbox* panel, which offers hundreds of algorithms from QGIS itself as well as third-party providers like GRASS and GDAL. The toolbox interface uses a tree view structure similar to a file browser, organizing algorithms into many general (e.g., cartography, raster analysis, vector analysis) and specialized categories (e.g., network analysis, 3d tiles). Although a search function is available, it remains difficult for users to fully understand the range of tools offered or their specific functionalities. Additionally, each third-party provider, including plugins, maintains its own set of categories that sometimes overlap with standard ones, preventing similar tools from being conveniently grouped together for easy comparison and assessment. While powerful, the toolbox exposes users to functions that are far beyond beginner level and makes it easy to apply the wrong process to the wrong data, which can lead to confusion or frustration without providing clear feedback or error handling.

Another aspect of QGIS that significantly influences the user interface and overall user experience is its plugin system. This system is highly versatile, allowing developers to enhance QGIS's core functionality by adding new tools, features, and algorithms. This facilitates rapid innovation and the sharing of new capabilities, empowering users to customize QGIS according to their specific workflows and creating a highly personalized GIS environment. However, the graphical interface design of plugins lacks standardization and is often inconsistent. While some plugins offer user-friendly and well-documented interfaces, others can be confusing or poorly supported. Additionally, managing plugin installation, updates, and dependencies can be technically demanding, posing a barrier for less experienced users, such as citizen scientists.

Overall, while QGIS offers extensive options for customizing the user interface, certain gaps remain that need to be addressed to support broader adoption by non-technical users. Therefore, efforts to simplify the interface are necessary.

3. Simplification of the QGIS Interface

To simplify QGIS, first, an analysis of the user needs is performed considering the target user group, which is non-technical people with limited geospatial data literacy and, in some cases (e.g., senior citizens), limited experience with desktop applications. It is assumed that users receive a basic introduction to core geospatial concepts, such as raster and vector data, map projections, and fundamental cartographic principles, prior to using QGIS, for example through a brief training session. It is also assumed that users possess basic familiarity with using desktop applications.

The identified needs are translated into the user stories below, which guide the design and development process:

S1. As a user, I want QGIS to open with a clean and simple interface, so that I can get started without feeling overwhelmed.

S2. As a user, I want to get brief tips or guidance the first time I use a feature, so that I can learn how to use it without needing a manual.

S3. As a user, I want to open map data from my computer or access it from online sources, so that I can view and work with geospatial information.

S4. As a user, I want to explore and use map layers in two dimensions like a paper map, so that I can easily understand the spatial patterns.

S5. As a user, I want to work with one map at a time, so that I stay focused.

S6. As a user, I want to view my data clearly on the screen, but I don't need to create printable maps or professional layouts.

S7. As a user, I want to use simple tools for exploring data, so that I'm not overwhelmed by complex functionality.

S8. As a user with basic map knowledge, I want to set or adjust the coordinate system if needed, so that my data lines up correctly.

S9. As a user, I want to add base maps like OpenStreetMap, so that I can see my data in context that I'm familiar with.

S10. As a user, I want to make simple charts from my data, so that I can better understand patterns or values.

Because the design is for non-technical users with limited knowledge on the capabilities of a modern desktop GIS, i.e. since users don't know what they don't know, the user stories are rather simple. Hence, considering these user stories, additional technical design constraints are introduced for better scoping of necessary simplifications, which are listed below. Related user stories are indicated in parentheses for each constraint:

C1. Default to a minimal interface with only essential toolbars and panels visible (S1).

C2. Include optional tooltips or guided popups to reduce reliance on external documentation or long tutorials (S2).

C3. Support for loading local vector and raster data files (e.g., Shapefile, GeoJSON, GeoTIFF) (S3).

C4. Support for web-based data access via OGC services (WMS, WMTS, WFS) (S3).

C5. No support for direct connections to spatial databases (e.g., PostGIS, MS SQL, Oracle) (S3).

C6. Only 2D vector (points, lines, polygons) and raster data are supported (S4).

C7. No support for 3D visualization, Z/M values, point clouds, meshes, or temporal data cubes (S4).

C8. No support for multiple map views for comparison (S5).

C9. No access to the layout manager or print layout tools (S6).

C10. No advanced cartographic styling (S6).

C11. No access to model builder, advanced spatial analysis tools, or advanced processing algorithms (S7).

C12. Allow basic CRS selection and data re-projection, keeping CRS options minimal and easy to understand (S8).

C13. Provide a simple interface to add common base maps, such as OpenStreetMap, Bing, Google Maps (S9).

C14. Enable easy creation and updating of basic plots (e.g., bar, pie, histogram) from attribute data without the need for external charting application (S10).

Based on the identified user stories and technical design constraints, a detailed analysis was conducted on all QGIS features, including menus, toolbars, panels, and processing algorithms, to identify non-essential and redundant components. This analysis focused on the following key aspects:

1. *Elimination of multiple access points for the same functionality.* In QGIS, it is common for a single function to be accessible through a menu item, a toolbar button, and a processing toolbar entry. While this approach is intended to improve accessibility, it often causes confusion and adds unnecessary complexity for new users. It can also complicate communication, such as during training sessions. In this study, all multiple access points are regarded as redundant, and only one access method is chosen to simplify function access, which is tool buttons.

2. *Elimination of multiple tools with identical or similar functionality.* In addition to offering native tools and algorithms for data analysis and processing, QGIS supports third-party processing providers. These providers are free to offer any processing capabilities, including those already available in QGIS. For example, the default QGIS installation includes powerful providers like GDAL and GRASS, which are well-established geoprocessing platforms with numerous tools that often overlap with QGIS's native tools. Although this can be advantageous by providing more advanced or efficient tools, it also adds unnecessary complexity for new users, complicating the selection of the most suitable option among similar choices. Because many third-party providers are closely integrated into QGIS, users often view them as standard QGIS tools, making multiple "standard" options confusing. This study tackles the issue by eliminating these redundant tools. If multiple tools offer similar functionality, the simpler one with the least complex features, yet still providing the minimum viable functionality, was chosen. This approach contrasts with the typical practice of selecting the most comprehensive tool with the greatest number of features. But the goal is to simplify the user experience rather than complicate it with additional technical details that are largely unfamiliar and irrelevant to the target audience. While advanced options are valuable for more precise geospatial analysis, they tend to overwhelm beginner-level users and cause confusion.

3. *Elimination of advanced tools and functionality.* To provide a simple user interface to perform basic GIS tasks, all advanced tools and functions should be removed. In this study, this is done by hiding such tools and functionalities.

Based on the findings of this assessment, the following actions have been identified as beneficial and essential for simplifying QGIS and enhancing its user experience for the target audience:

- *No menu bar.* All menu items should be removed and necessary features linked to menu items should be provided as tool buttons.

- *Less toolbars.* The number of toolbars should be reduced to two, one for core data access, query and visualization functions, and another one for editing purposes. Besides removing unnecessary toolbars, common functions that are provided by multiple tool buttons (e.g., zoom in, zoom out, zoom full, zoom to selection, etc.) should be grouped and made available through dropdown tool buttons, which occupy less space and provide a compact yet efficient interface (Figure 2).

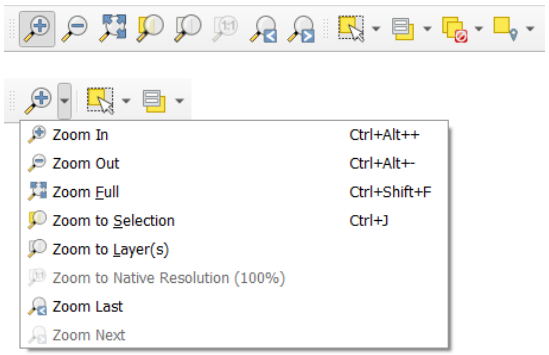

Figure 2. Example compact dropdown tool group

- *Less panels.* Only the overview and layer panels should be made visible by default. The rest should be hidden and became visible only if they are needed, i.e. a related function is requested. Besides enabling more space for the map canvas, less panels also facilitates more focused user experience by removing unnecessary distractions.

- *Fixed layout of the toolbars and panels.* The locations of GUI components should be fixed and the capability of the user to change the layout of the working environment should be disabled, preventing to move or float toolbars and panels. This is to ensure a consistent and identical user experience among the users, which is especially important when, e.g., training non-technical users.

- *No processing toolbox.* All essential processing algorithms should be made accessible via dropdown tool buttons added to the main and editing toolbars. This enables a single, standardized method to access processing functions, and increases the space available for the map canvas.

- *Limited set of data sources.* The ability to read complex geospatial data, such as mesh and point clouds, and to connect to local and remote databases (e.g., PostgreSQL, Oracle), tile sets (e.g., vector, raster, 3D), and specialized geodata servers (e.g., ArcGIS Server) should be disabled. These advanced data sources are often unnecessary for novice users, particularly in citizen science applications. Instead, data sources should include standard raster and vector formats, OGC web services (WMS, WFS, WCS), and formats relevant to citizen science such as SensorThings for sensor and IoT data and STAC for cloud-native data access.

- *Less processing algorithms.* The algorithms that are classified as advanced, duplicate, or unnecessary should be removed. Based on our analysis, these should include SQL, Z/M, database, TIN, mesh, tile, curve, GPS, cartography, random, fuzzify functions, as well as modeler tools and functions provided by GRASS and PDAL as summarized in Table 1. A comprehensive list of all algorithms available in QGIS, along with justifications for their inclusion or removal, is provided in the documentation section of the QGIS Light code repository[1]. Common reasons for excluding certain algorithms include limited applicability, availability of equivalent functionality in other parts of the interface (e.g., attributes table, raster calculator), duplication with similar algorithms, reliance on external applications (e.g., for displaying output), and relevance primarily to advanced use cases.

Category	Examples of unnecessary algorithms
Cartography	Align points to features, Combine style databases, Export atlas layout, etc.
Curve	Convert to curved geometries, etc.
Database	Execute SQL (SpatiaLite, PostgreSQL), Export to SQL (PostgreSQL), etc.
Fuzzify	Fuzzify raster (gaussian, linear, etc.)
GPS	Convert GPS data, Download GPS data, etc.
GRASS	General (g.*), Imagery (i.*), Raster (r.*), Vector (v.*) functions
Mesh	Export contours, Export mesh edges, Rasterize mesh dataset, Surface to polygon, etc.
Modeler Tools	Create directory, load layer into project, set project variable, feature filter, etc.
Point cloud	Point cloud data management, Point cloud conversion, Point cloud extraction functions
Random data generation	Create random raster layer (binomial, exponential, etc.), Random points functions
Tile	Download vector tiles, Generate XYZ tiles (Directory, MBTiles), etc.
TIN	TIN interpolation, etc.
Z/M	Set Z value, Drop M/Z values, etc.

Table 1. A short list of unnecessary algorithms.

- *Additional features.* To enhance the user experience, the following features should be added to the interface by default via existing plugins:

  Plotting functions should be replaced with *Data Plotly*, a QGIS plugin that supports the creation of interactive charts with features like zooming and hover-based information display (Ghetta, 2024). It allows users to easily modify chart settings, such as type, colour, and axes, and offers a wider variety of plot types than QGIS. Additionally, it integrates directly with the map canvas, enabling dynamic updates based on user-selected features and eliminating the need to open external files to view plots, as required by QGIS's default approach. This significantly enhances the user experience. Plots can also be exported in a range of image and vector formats, with the latter supporting easy post-processing and improved interoperability.

  Common base maps should be provided by using the *QuickMapServices* plugin (NextGIS, 2024), which integrates with popular map providers and offers a wide selection of base maps that can be added as layers easily.

[1] https://github.com/ITC-CRIB/qgis-light/tree/main/docs

4. QGIS Light Plugin

To provide access to a simplified and streamlined user interface for QGIS by implementing the necessary actions listed in the previous section, *QGIS Light* was developed as a QGIS plugin. The plugin utilizes customization options available in QGIS and, when necessary, interacts directly with Qt to enable advanced customizations not natively supported by QGIS. Users can easily adjust the applied simplifications by editing a configuration file to disable specific simplifications or enable new ones. The plugin is available on the QGIS plugin repository, making it easy to install through the QGIS plugin manager[2]. The source code is open access under the GPL 3.0 license (Girgin, 2024), and the open-source code repository is hosted on GitHub to facilitate collaboration, which is available at https://github.com/ITC-CRIB/qgis-light. For long-term preservation, the source code is also archived on Zenodo[3].

Once installed and enabled by using the plugin manager, QGIS Light does not immediately simplify the GUI. Instead, it adds a QGIS Light button with a plain QGIS logo to the *Project* toolbar. Clicking this button or selecting the *Toggle QGIS Light* option in the *View* menu activates the simplified interface (Figure 3). QGIS remembers this setting and automatically launches the simplified GUI in future sessions unless it is disabled. To disable the plugin, the colourful QGIS tool button on the *Main* toolbar of the simplified interface can be utilized.

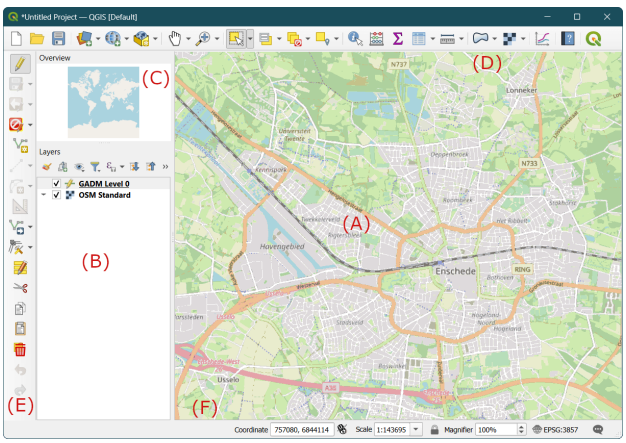

Figure 3. QGIS Light user interface, a) Map canvas, b) Layers panel, c) Overview panel, d) Main toolbar, e) Editing toolbar, f) Status bar.

Once activated, QGIS Light performs the following simplifications in the order listed:

1. Menu bar is hidden.
2. Contextual menu is disabled. This prevents toolbars and panels to be customized by the user.
3. A record of the currently visible toolbars and their positions is created, and then the toolbars are hidden.
4. Simplified toolbars are created and then displayed.
5. A record of all panels is created, capturing their location, properties (e.g., dockable, movable) and visibility status.

[2] https://plugins.qgis.org/plugins/qgis-light/
[3] https://doi.org/10.5281/zenodo.13844843

6. Panels of the simplified GUI are layout and made visible, and the other panels are hidden.

7. Data source managers and data item providers are set up.

8. Status bar is set up.

When QGIS Light is activated, information on the current state of the toolbars and panels are saved in the QGIS registry. This allows the GUI to be restored to its original state in the future, ensuring consistency across sessions. When the plugin is deactivated, the interface is reverted by removing the simplified toolbars and restoring and enabling components hidden or disabled by the plugin. Due to limitations with data source and data item providers, restarting QGIS is necessary to fully restore the original list of providers. A notification is shown to inform the user of this requirement.

The QGIS Light user interface consists of six components: the map canvas, layers panel, overview panel, main toolbar, editing toolbar, and status bar (Figure 3). The map canvas and the two panels are the same as in standard QGIS, with the difference that their positions are fixed to the left and right of the main window, respectively, though their sizes can be freely adjusted. The status bar at the bottom is largely unchanged, except that widgets related to map rendering, such as rotation control and rendering checkbox, have been removed. The main toolbar, positioned at the top, contains tools for project management, adding and creating data layers, navigating maps, selecting and measuring features, accessing attribute data, processing data, plotting, and getting help. This toolbar also features the QGIS Light toggle button, enabling users to switch back to the default interface. The editing toolbar, located on the left, includes tools for managing editing sessions (e.g., start editing, save, rollback), modifying and transforming features (e.g., cut, paste, rotate, scale), creating features (e.g., polygon, circle, rectangle), and editing attribute data. To maintain a compact design, tools are organized by function and, when appropriate, grouped into drop-down menus in both toolbars. Only the most commonly used tools are shown as individual buttons.

Most of the simplifications listed above can be customized by editing the configuration file (*config.json*) located in the plugin directory. The configuration file is divided into five sections: *toolbars*, *panels*, *algorithms*, *providers*, and *statusbar*. A simplified example configuration file is shown in Figure 4, where certain parts are replaced with ... due to space limitations.

The *toolbars* section defines which toolbars will appear in the simplified GUI. Each toolbar is specified by a unique identifier (e.g., *mMainToolBar*), which must not conflict with those used by QGIS or other installed plugins. Each toolbar entry includes a title, a location (e.g., top, left), and a list of items to display. An item typically represents a tool from an existing toolbar, referenced by combining the original toolbar id and the tool's action id with a colon (e.g., *mFileToolbar:mActionNewProject*). When multiple such identifiers are listed in an array, they are grouped into a drop-down button, with the first item shown as the default. Similarly, groups of algorithms defined in the algorithms section can also be presented as drop-down buttons. A *separator* item can be added between tools, tool groups, or algorithms to visually separate them. By default, the configuration file includes two toolbar definitions, one for the main toolbar and another one for the editing toolbar. Additional toolbars can be added, or existing ones can be modified to further customize the simplified interface.

The *algorithms* section allows processing algorithms provided by QGIS to be organized into tool groups that can be added to toolbars. Each algorithm group is identified by a unique id and includes an icon along with a list of algorithm items. Like tool items, algorithm items are identified by two-part identifiers: one part designates the processing provider, and the other specifies the algorithm itself (e.g., *native:buffer*). Currently, algorithms are categorized into two groups as raster or vector based on the data type they operate on. All core raster and vector processing algorithms provided by QGIS either natively or by using GDAL are included and separated by section headings for easier navigation. The lists represent an initial selection and may be revised in the future based on community feedback.

The *panels* section enables specifying which panels will be available in the simplified interface, along with their placement and initial visibility. Any panels not included in this section are hidden by the plugin, and their associated functionalities, such as tools and algorithms, are also disabled. The *providers* section allows data source and data item providers enabled in the simplified interface to be specified. Data source providers are components that enable QGIS to connect to and read data from various data sources, whereas data item providers handle how the data is represented, managed, and interacted with once loaded within the QGIS environment. They can be enabled by adding their identifiers to the list of data sources or data items in this section. Finally, the *statusbar* section enables certain widgets to be disabled. QGIS Light features utility methods to retrieve the ids of toolbars, tool actions, algorithm providers, algorithms, data source providers, data item providers, and status bar widgets to facilitate easy configuration.


```
{
  "toolbars": {
    "mMainToolBar": {
      "title": "Main Toolbar",
      "area": "top",
      "items": [
        "mFileToolBar:mActionNewProject",
        "mFileToolBar:mActionOpenProject",
        ...
      ]
    },
    "mEditingToolBar": {
      "title": "Editing Toolbar",
      "area": "left",
      "items": [
        "mDigitizeToolBar:mActionToggleEditing",
        [
          "mDigitizeToolBar:mActionSaveEdits",
          "mDigitizeToolBar:mActionSaveAllEdits"
        ],
        ...
      ]
    }
  },
  "algorithms": {
    "raster": {
      "icon": ":/images/themes/default/mIconRaster.svg",
      "items": [
        "gdal:viewshed",
        "native:hillshade",
        ...
      ]
    },
    "vector": {
      "icon": ":/images/themes/default/mIconVector.svg",
      "items": [
        "native:buffer",
        "native:clip",
        ...
      ]
    }
  },
  "panels": {
    "Overview": "fixed:left",
    "Layers": "fixed:left",
    "DataPlotly-DataPlotly-Dock": "hidden:right",
    ...
  },
  "providers": {
    "data_sources": [
      "delimitedtext",
      "gdal",
      ...
    ],
    "data_items": [
      "files",
      "GPKG",
      ...
    ]
  },
  "statusbar": {
    "LocatorWidget": false,
    "mRotationLabel": false,
    "mRotationEdit": false,
    "mRenderSuppressionCBox": false
  }
}
```


Figure 4. An example configuration file (shortened).

https://doi.org/10.5194/isprs-archives-XLVIII-4-W13-2025-127-2025

5. Additional Usability Challenges in the QGIS Interface

Although this study and the QGIS Light plugin primarily focus on simplifying QGIS's user interface elements, the assessment of QGIS components revealed additional insights aimed at enhancing a more consistent and user-friendly experience. Some observations highlighting inconsistencies and areas for potential improvement are outlined in this section below. It is important to note that these points are not limited to non-technical users but are generally relevant for all users:

- Terminology is often inconsistent across algorithms that perform similar tasks, and even within the same algorithm when describing parameters. For example, terms like *create*, *calculate*, and *compute* are used interchangeably to refer to the generation of an enclosing circle.

- Multiple algorithms exist to perform similar tasks on different geometries. For example, the *Convert Geometry Type*, *Polygonize*, and *Lines to Polygons* algorithms all convert vector geometries and can be used to transform lines into polygons. However, the distinctions between these algorithms are unclear, and there is no straightforward guidance on which one to use for a particular situation, especially if their functions differ.

- Some algorithms that serve similar or complementary purposes offer different sets of features. For example, the *Remove Duplicates by Attribute* algorithm allows saving duplicate records to a separate file, whereas the *Remove Duplicate Geometries* algorithm, which performs a similar operation based on geometries rather than attributes, lacks this option. This results in inconsistency between the algorithms, which are expected to belong to the same functional group.

- Some algorithms have very similar names but carry out their tasks in different ways. For example, *Fill NoData* algorithm provided by GDAL uses interpolation to fill missing data cells, while the native *Fill NoData Cells* algorithm fills them with a constant value. This distinction is not immediately clear from the names as they appear in the processing toolbox.

- Some algorithms share the same name and perform the same core task but offer different sets of parameters. For example, the native *Hillshade* algorithm includes options for Z factor, horizontal angle (azimuth), and vertical angle, while the GDAL version provides additional parameters such as scale, compute edges, the Zevenbergen-Thorne formula, combined shading, and multidirectional shading. Moreover, GDAL uses different terminology, referring to the Z factor as vertical exaggeration and the angles as the azimuth and altitude of the light. In general, GDAL tools tend to offer more configuration options than their native counterparts. It is not immediately obvious why both versions are available for the same task, especially when one effectively encompasses the other.

- Some algorithms are primarily intended for use within the model builder, such as the *Drop Fields* tool. These are generally not needed for direct, interactive use, as their functions can often be performed more easily through the user interface. For example, in case of *Drop Fields*, by removing fields directly in the attribute table. Displaying these algorithms by default can lead to unnecessary clutter in the processing toolbox.

- Some raster algorithms are required solely because certain operations are not supported by the *Raster Calculator*. For example, the *Round Raster* algorithm rounds cell values to a specified number of decimal places. This is a basic numerical operation that one would reasonably expect to find in the raster calculator, which is specifically designed for performing numerical operations on raster datasets. However, this functionality is currently missing.

- Some algorithms are entirely covered by more versatile counterparts. For example, the *Difference* algorithm functions identically to the *Difference (Multiple)* algorithm when only a single overlay layer is used, with the sole distinction being support for in-place processing in case of the former. It is not immediately clear why in-place processing is not supported by the latter and it's worth noting that in fact it is a rarely supported feature, which is not always functioning properly.

- Unlike native algorithms, GDAL algorithms lack embedded help content, even though detailed information is available in the online documentation. For example, the *Aspect* algorithm provided by GDAL has a comprehensive explanation of each parameter on its online help page, but within the algorithm dialog, parameters are only listed by name without any descriptions.

- Similarly, the embedded help for some native algorithms is limited, despite detailed explanations being available in the online documentation. *The Export to SpatiaLite* algorithm is one such example. In some cases, there are also inconsistencies between the embedded help and the online documentation, which requires further attention.

- Inconsistencies exist between algorithms that perform the same task using different methods. For instance, the *Concave Hull* algorithm only accepts point layers and produces a single polygon, with no option for grouping. In contrast, the *Concave Hull (K-Nearest Neighbor)* algorithm also accepts polygon layers by automatically extracting points and supports generating multiple concave hulls based on attribute grouping. Similarly, while the *Convex Hull* algorithm generates a convex hull for each individual feature, the *Minimum Bounding Geometry* algorithm can create a single convex hull for all features, similar to the behaviour of the *Concave Hull* algorithm. Although these algorithms are closely related, their differing inputs and behaviours can lead to confusion.

- Outputs generated as external files often disrupt the user experience because they don't appear directly within the QGIS window. Instead, users are typically given only a cryptic path to a temporary file, which they must open manually. Unfortunately, this is how all plotting and statistical summary algorithms function, despite being commonly used tools for exploratory data analysis.

6. Conclusions

The primary goal of QGIS Light is to provide a simplified entry point for non-technical individuals to engage in spatial data analysis. While non-technical people are often involved in data collection part of citizen science activities through specialized applications, they are typically excluded from data analysis and evaluation, which is usually handled by experts. By offering a simplified GIS interface, we can make spatial analysis more accessible to such people without sacrificing the software's

powerful capabilities. This approach can serve as a stepping stone, allowing users to gradually transition to the standard GIS interface and advanced features, fostering GIS community growth besides participation in citizen science activities. In fact, a simple interface might be useful for anybody that requires core data visualization, editing, and analysis functionality, and can facilitate education, capacity development, and even professional activities. Therefore, the plugin has potential applications that go beyond its original focus on citizen science. It can also support secondary and higher education, lifelong learning, and private sector use, for example, by providing broad access to spatiotemporal analysis capabilities for large groups of employees. The plugin's FOSS nature and flexible configuration options allow the simplifications to be customized to meet the specific needs of other user groups and domains.

To encourage wider adoption of the plugin, a simplified installation process, ideally integrated with the QGIS installation, would be highly beneficial. This would allow non-technical users to start using the QGIS Light interface immediately, avoiding the current two-step process where they first install QGIS and then add QGIS Light as a plugin, often exposing them to the standard interface in the meantime. Another potential improvement is to provide streamlined help content tailored specifically for the simplified interface. Currently, QGIS help directs users to online documentation designed for the standard full interface, which can be confusing since the simplified interface does not support all the features. Automatically generating help content for the simplified interface based on the standard QGIS documentation could offer clear, user-friendly guidance while ensuring maintainability. Although it might require source-code changes, adding the ability to hide or remove advanced options (e.g., feature filtering), advanced settings (e.g. number of threads), and batch processing from algorithm dialogs would create a cleaner and more straightforward experience for novice users. Notably, advanced settings are not supported by all algorithms, and there is no indication of their actual availability, which can be confusing even for experienced GIS users. Finally, including help content for algorithms from third-party providers, like GDAL, as well as for some native algorithms whose documentation exists online but is not accessible within the application, would greatly enhance the overall user experience.

During this study, we identified several additional factors that hinder a better user experience, including inconsistent terminology, similarly named tools with differing parameters, tools with nearly identical names performing distinct tasks, and tools that could be consolidated. Our findings suggest that critically reviewing existing user interface elements and processing algorithms to streamline them into a more refined and standardized experience could improve usability of QGIS. This approach may also serve as a model for simplifying other GIS software. To support such an initiative, more in-depth user interface and user experience research focusing on both technical and non-technical QGIS users would be valuable. Such research could also help refine the current selection of core components in the simplified interface, which is primarily based on our own experiences and technical design constraints derived from a limited set of user stories. While the feedback received so far suggests that the selection has been effective, the evaluation remains somewhat subjective. As a direction for future work, we recommend a community-driven process involving design and usability experts, enabling the systematic integration of both quantitative and qualitative user feedback. We hope that QGIS Light serves as the first step toward such an initiative.

Author Contributions

Serkan Girgin originated the idea, performed the needs assessment, designed the simplification process, and developed the QGIS Light plugin. Jay Gohil and Indupriya Mydur contributed to analysing the QGIS components selected for simplification.

References

Arias de Reyna, M., Simoes, J., 2016: Empowering citizen science through free and open source GIS. *Open Geospatial Data, Software and Standards*, 1, 7. doi.org/10.1186/s40965-016-0008-x.

Bonney, R., Shirk, J.L., Phillips, T.B., Wiggins, A., Ballard, H.L., Miller-Rushing, A.J, Parrish, J.K., 2014: Next steps for citizen science. *Science*, 343, 1436-437. doi.org/10.1126/science.1251554.

Coetzee, S., Ivánová, I., Mitasova, H., Brovelli, M. A., 2020: Open geospatial software and data: a review of the current state and a perspective into the future. *ISPRS International Journal of Geo-Information*, 9(2), 90. doi.org/10.3390/ijgi9020090.

Conrad C.C., Hilchey K.G., 2011: A review of citizen science and community-based environmental monitoring: issues and opportunities. *Environmental Monitoring and Assessment*. 176, 273-91. doi:10.1007/s10661-010-1582-5.

Ghetta, M., 2024: Data Plotly. Software, Version 4.2.0, https://github.com/ghtmtt/DataPlotly (24 October 2024).

Girgin, S., 2024: QGIS Light. Software, Version 0.1.1. Zenodo. doi.org/10.5281/zenodo.13844843 (26 September 2024).

Haklay, M., 2013: Citizen science and volunteered geographic information: overview and typology of participation. In: D. Sui, S. Elwood, M. Goodchild (Eds.) *Crowdsourcing Geographic Knowledge*, 105-122. Springer. doi.org/10.1007/978-94-007-4587-2_7.

Kocaman, S., Saran, S., Durmaz, M., Kumar, A.S. (Eds.), 2022: Citizen science and geospatial capacity building, *ISPRS International Journal of Geo-information (special issue)*, ISBN 978-3-0365-3714-6, doi.org/10.3390/books978-3-0365-3714-6.

NextGIS, 2024: QuickMapServices. Software, Version 0.19.36, https://github.com/nextgis/quickmapservices (19 November 2024).

QGIS.org, 2025: QGIS 3.40 Geographic Information System User Guide. https://docs.qgis.org/3.40/en/docs/user_manual/ (25 February 2025).

Raga, F., 2019: Customize ToolBars. Software, Version 1.11, https://github.com/All4Gis/CustomToolBar (19 March 2019).

Skarlatidou, A., Hamilton, A., Vitos, M., Haklay, M, 2019: What do volunteers want from citizen science technologies? A systematic literature review and best practice guidelines. *Journal of Science Communication*, 18(1), A02. doi.org/10.22323/2.18010202

The Qt Company, 2018: Qt - Cross-platform Application and UI Framework. Software, Version 5.12, https://qt.io (13 August 2018)