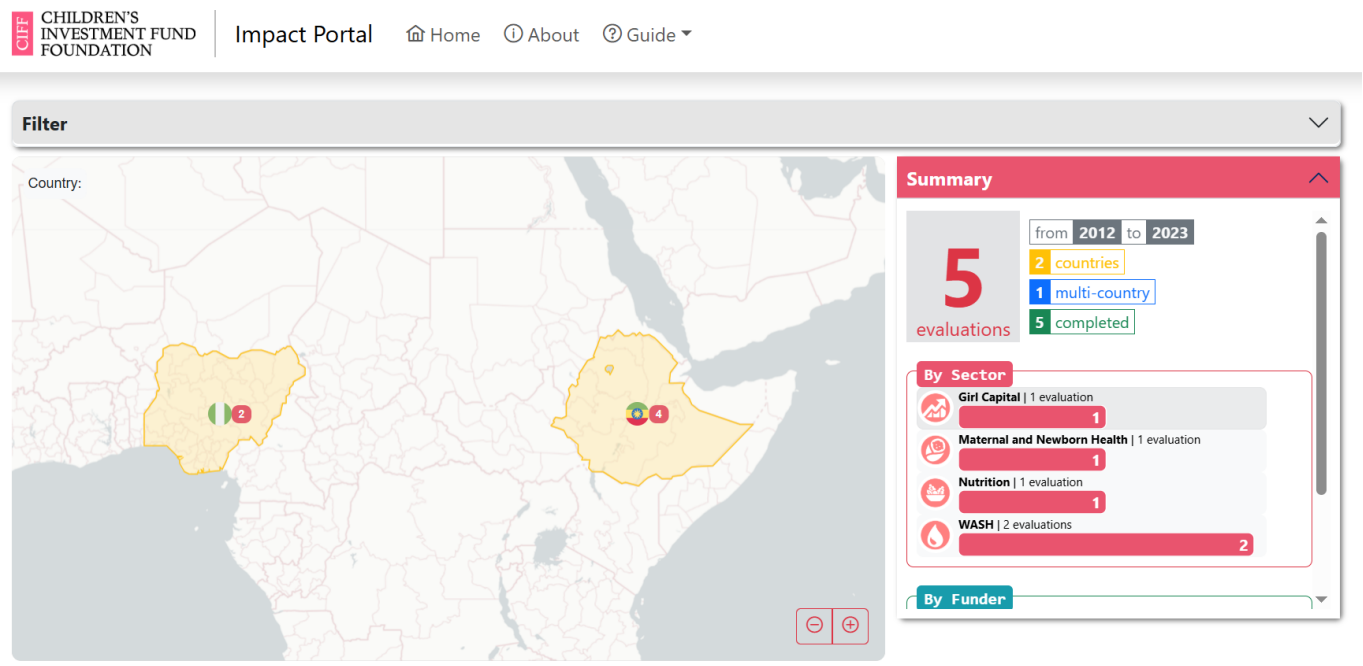
Impact Evaluation Portal

by Rhorom Priyatikanto (rp1y21@soton.ac.uk)



To support impact evaluation performed by Children Investment Fund Foundation (CIFF), we developed a web application portal that includes an interactive map and data filtering. The portal acts as a showcase of investments/evaluations by CIFF happening in several countries around the Globe. It displays evaluations at both country- and subnational-level. Following the richness of information related to the listed investments/evaluations, dynamic overlays of the data on the map is implemented so that those information can be grasped easily.

App Name	**CIFF Impact Evaluation Portal**
Customer	CIFF (Children Investment Fund Foundation)
Project lead	Carla Pezzulo
Development Team	Rhorom Priyatikanto, Maksym Bondarenko
	SDI Worldpop, University of Southampton
Repository	https://github.com/ciffitsupport/impact_evaluation
Web URL	https://impact.ciff.org

About this file

The purpose of this file is to provide overview, background information, and setup information of the project. If you have joined this project as a part of the development team, please ensure this file is up to date.

Note : Any dependencies added/modified to this project which affect the running of the code in this git repository must be listed in this file. All developers must ensure that the instructions mentioned in this file are

sufficient to enable a new developer to obtain a executable copy of the latest code in this repository, without involvement from any other human assistance.

Table of contents

Features

Several functionalities are activated to ensure interactivity, informativeness, and efficacy of the portal. Firstly, the main functionality is intuitive filters to narrow down displayed evaluations based on various criteria (country, sector, target population, primary outcome, and evaluation status). Secondly, listing programmes associated with a selected country. Form-based filtering is linked to the map so that country selection can be performed through clicking a particular country on the map or selection through form. In case of multi-country investment/evaluation, all countries relevant to the programme can be displayed on the map. Lastly, more extensive information (programme description, implementation years, results, etc.) for a particular programme can also be showcased in a neat format.

Responsive, reactive, and user friendly are three main characteristics of the aimed portal. Thus, we developed the portal using React-Vite as the framework. This framework was integrated with React-Router to enable deep linking and routing of the portal. User's experience and history are recorded in a structured URL so that sharing and tracking can be done. Responsive layout and components were constructed using React-Bootstrap with some additional customization based on the CIFF's brand sheet. Utilizing this library, the portal adapts flawlessly into different screen sizes, from smartphones to desktops. React-Leaflet becomes the backbone for interactive mapping of the data, supported by several publicly available basemaps. The high-level architecture of the portal and technical documentation associated can be found in the Architecture section. In terms of user friendliness, guidance and notes are provided on the portal. General description about the portal, its standard operating procedures, and even a glossary of terms and abbreviations are easily accessible from the header menu.

Considering the needs of periodic updates on the overlaying data, we use JSON data structures to encapsulate both tabular data associated with the CIFF investments/evaluations and the geographic data defining administrative boundaries and points of interest.

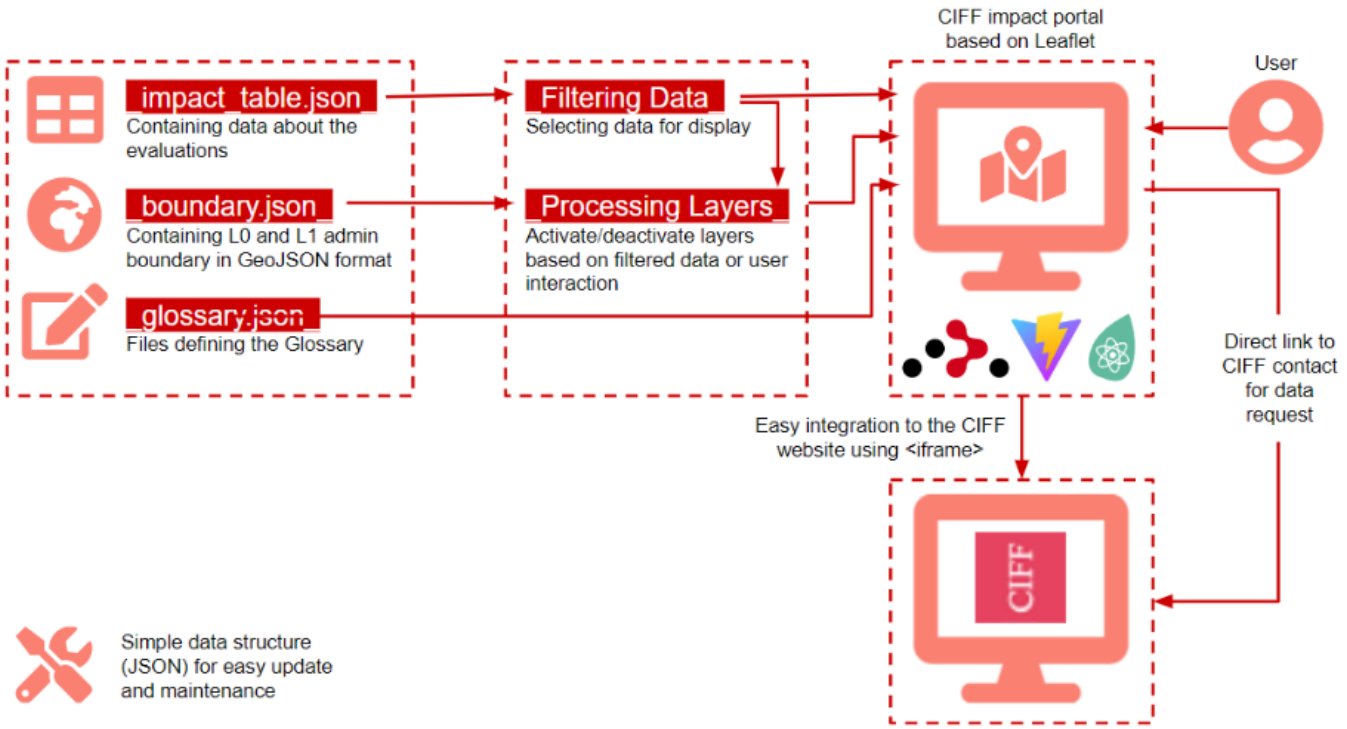
Dependencies

This portal mainly depends on the following javascript libraries:

- React 18.2.0 for reactive application development
- React-router 6.21.0 for routing and deep-linking feature
- React-leaflet 4.2.1 for interactive mapping in react
- React-bootstrap 2.9.1 for responsive layout and styling
- Primeicons 6.0.1 as icons library
- Html-react-parser 5.0.11 for parsing text with HTML-format
- React-circle-flags 0.0.23 for visualisation
- Formik 2.4.6 for handling forms
- React-simple-wysiwyg 3.2.2 for WYSIWYG editor

Architecture

CIFF Impact Evaluation Portal aims to display and map the investments by CIFF all over the world and the associated impacts. Users can interact with the portal by filtering the data and selecting the country of interest where CIFF investments and evaluations exist. This portal is based on React. It has a responsive design that enables users to access the portal using various devices. Seamless integration of the portal to the main CIFF website became a major consideration in the portal design and development. It uses the standard CIFF color scheme as in the main CIFF website. Its architecture and data structure are relatively simple so that further update and development of the portal will not be head scratching.



Source files

Directory tree

```
ciff_impact_portal
|-- dist
|-- public
|-- src
|   |-- data
|   |   |-- boundary.json
|   |   |-- glossary.json
|   |   |-- impact_table.json
|   |-- App.jsx
|   |-- config.jsx
|   |-- Edit.jsx
|   |-- Evaluation.jsx
|   |-- Graphic.jsx
|   |-- index.css
|   |-- main.jsx
|   |-- MainApp.jsx
|   |-- Map.jsx
|   |-- MapSmall.jsx
|   |-- Pages.jsx
|   |-- PanelFilter.jsx
|   |-- PanelInfo.jsx
|   |-- utils.jsx
|-- index.html
|-- package.json
|-- README.md
```

File description

File name	Description
`dist/`	Default folder for the optimized scripts and files to deploy. Command `vite build` will do the optimisation and fill this folder with the resulting files.
`public/`	Directory for images used in the portal.
`index.html`	The main HTML file for the portal. It calls all required JavaScript libraries and CSS stylesheets. The main script (src/main.jsx) is sourced by this main file.
`package.json`	Basic information about the app, its dependencies, and script aliases to build, test, and deploy the app. App name, version, and homepage are defined in this file.
`vite.config.js`	Configuration and settings for Vite are stored in this file. Normally, we do not need to change anything in this file.
`src/config.jsx`	Containing some variables to configure the app.
`src/index.css`	The CSS file for rendering the app.
`src/main.jsx`	The main javascript which renders all react components associated with the app.

File name	Description
src/App.jsx	All react components used in the portal are defined and returned in this file.
src/MainApp.jsx	This file defines the main part of the map, which are filter panel, map panel, and info panel on the right.
src/Edit.jsx	Defining component for editing evaluation entry or adding a new one.
src/Evaluation.jsx	Defining the information related to a specific evaluation.
src/Graphic.jsx	Visual summary of the listed evaluations is governed by this source file.
src/Map.jsx	This javascript dictates how the map is displayed and how it interactively behaves.
src/MapSmall.jsx	Defining evaluation-specific map that is called in Evaluation.jsx and PanelInfo.jsx.
src/Pages.jsx	Contents displayed in the 'modal' when the user clicks 'About', and 'Guide' menu on top of the page.
src/PanelFilter.jsx	Defining the filtering functionality of the app.
src/PanelInfo.jsx	Dictating how the info panel (right to the map) looks like.
src/utils.jsx	Containing basic functions required in the app.

Data structure and update

There are two main data files to be sourced by the portal. Both are stored as JSON and can be easily updated.

Boundary file (src/data/boundary.json)

Geographical data defining the administrative boundary of countries (L0) and regions (L1) relevant to the CIFF impact evaluation is stored in GeoJSON format with specifically defined columns. This is a standard and widely-used format so that update and extension will not be difficult. Prior knowledge on this data format will be useful for further update and development.

boundary.json contains geo-coordinates of the administrative boundary of the countries (L0) and regions (L1) which are accessible through the portal. Country, CountryID, Region, and RegionID are keys/columns that need to be synchronized with impact_table.json. CountryID is basically similar to Country, but with whitespace removed. The same applies to RegionID. Level should identify the administrative level, either ADM_0 or ADM_1. Lon and Lat specifies the centroid of the boundary over which marker will be placed. geometry defines the boundary itself. It may contain POLYGON or MULTIPOLYGON objects.

In case more countries or regions are added to the list of evaluations, boundary.json needs to be extended accordingly. Most of the boundaries listed in the portal were sourced from Geoboundaries or other sources. Boundary data in GeoJSON format can be downloaded from this site. If a different source is preferred and a different format (e.g., shapefile) is provided, then the boundary data needs to be converted into GeoJSON format prior to the addition to the boundary.json. For this purpose, an online converter will absolutely be useful. The acquired boundary data needs to be harmonized such that the properties or columns are similar to that of boundary.json.

```
{
    "type": "FeatureCollection",
    "crs": {
        "type": "name",
        "properties": {
            "name": "urn:ogc:def:crs:OGC:1.3:CRS84" }
            },
    "features": [
    {
        "type": "Feature",
        "properties": {
            "FID": 1,
            "Country": "Bangladesh",
            "CountryID": "Bangladesh",
            "CountryISO": "bd",
            "Region": " ",
            "RegionID": " ",
            "Level": "ADM_0",
            "Lon": 89.889,
            "Lat": 23.674
            },
        "geometry": {
            "type": "MultiPolygon",
            "coordinates": [ [ [
                [ 92.314, 20.666 ],
                [ 92.314, 20.666 ], ...
                ] ] ]
            }
    }
    ]
}
```


Impact table (`src/data/impact_table.json`)

Extra attention is needed for important keys (or column names) that are called in the script. Some keys are interrelated as the activation/deactivation of boundary layers in the map is based on the filtered impact data.

In `impact_table.json`, each JSON object is associated with a particular evaluation as indicated by its `EvaluationID`. `Country` contains the country name(s) where the evaluation is happening while `CountryID` contains the same name(s), but whitespace(s) removed. If there are multiple countries, then the `Multi` should be "Yes".

`Image` column defines the path to image file representing the evaluation. The image file is kept in `./public/` directory that is regarded as root for images, e.g., image path of `./image-sample.png` refers to `./public/image-sample.png`.

`Region` may be empty or containing a list of the regions associated with the evaluation. Values of `Country`, and `Region` should be consistent with the ones in `boundary.json`. To be noted that `Sector`, `Type`, `Funder`,

and `Years of Investment` are used for data filtering, altogether with `Country`. So, standardisation of their values is essential. Lengthy list of possible values in the filter needs to be avoided as it may confuse the user.

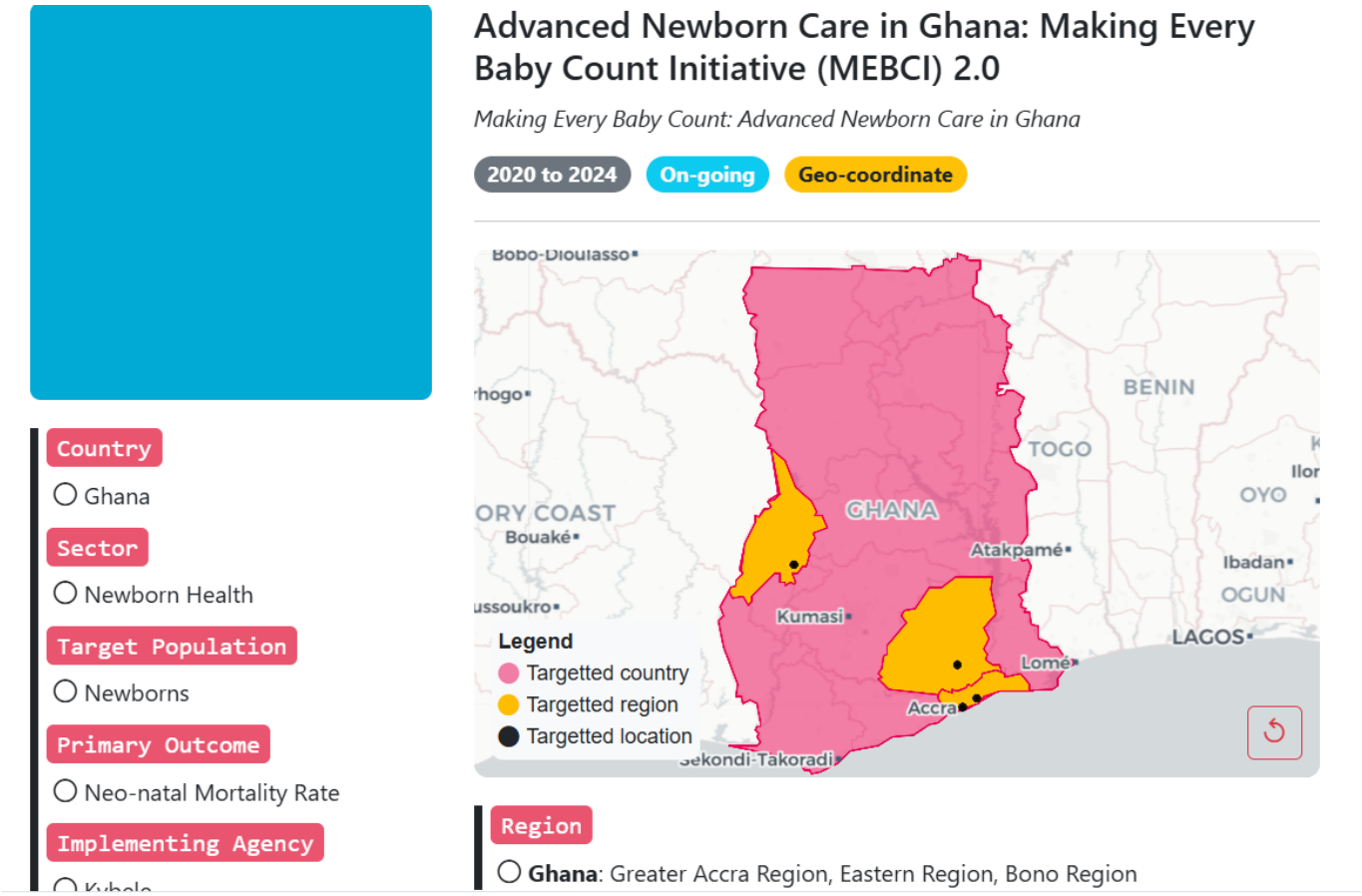
```
{
        "EvaluationID":4,
        "Investment Name":"ENAT-Enhancing Nutrition and Antenatal Infection
Treatment for Maternal and Child Health",
        "Proposed Public Title":"Enhancing Nutrition and Antenatal Infection
Treatment for Maternal and Child Health",
        "Image":"./image-sample.png",
        "Status":"Completed",
        "Funder":"CIFF",
        "Country":["Ethiopia"],
        "Region":["Amhara", "Oromia"],
        "Coordinate":"[[0,0],[0,0.1]]",
        "Sector":"Maternal and Newborn Health",
        "Primary Outcomes":"Low birth weight (LBW)",
        "Type":"Impact evaluation",
        "Implementing Agency":"Jhpeigo",
        "Evaluation Agency":"MELA",
        "Years of Investment":"2018, 2019, 2020, 2021, 2022",
        "Investment Amount":"$3,550,000",
        "Study Design":"Cluster randomized control trial",
        "Programme Description":"",
        "Areas of Programme":"",
        "Impact Statement":"",
        "Link to Publication":"",
        "Multi":false
},
```

For completed investment/evaluation, published report/work can be available. The link to such publication is stored in `Link` column.

The `Study Design` ... `Data Availability` may contain plain text or HTML-like element (HTML tags can be applied) that will be displayed in table as detail information about the evaluation. If the evaluation includes geo-coordinates, `Coordinate` will contain array containing Longitude-Latitude pairs as its value.

```
"Coordinate":"[[LON1,LAT1], [LON2,LAT2], [LON3,LAT3], ...]"
```

To modify the content of this file, just edit the value according to the rules stated before. Do not forget double-quote in each value. To add a new evaluation item, just add a new JSON-like object with the same structure. Make sure that the country or region name is synchronized with one in `boundary.json`.



Alternative edit

To edit an evaluation entry's JSON record according to the defined standard, use the editing form accessible in this https://impact.ciff.org/ciff_impact_portal/#/edit.

Existing records/evaluations are listed in a table. By clicking a particular record, an editable form will open. Modify the field values, referring to the short info provided for each field to ensure correct input.

To create a new record, click `Empty Form` at the top.

To download the JSON file for the currently edited entry, click `Download` at the bottom. To download a JSON file containing all records, including your updates, click `Merge and Download`.

Finally, append or merge the downloaded file with `src/data/impact_table.json`. Redeployment will be required to update the github page. See Deployment setup for more instruction.

Development setup

1. Install Node.js and npm.

2. Setup a working directory and install required libraries

```
mkdir webApp
cd webApp
npm -i react vite react-router ...
```

3. Clone repository

```
git clone https://github.com/rhorom/ciff_impact_portal
cd ciff_impact_portal
```

4. Start development server to preview file changes: `npm run dev`

Deployment setup

1. Configure the package by modifying `package.json`

```
{
    "name": "ciff_impact_portal",
    "homepage": "https://impact.ciff.org",
    "private": true,
    "version": "0.2.0",
    "type": "module",
    ...
}
```

2. Build the app: `npm run build`

3. App preview: `npm run preview`

4. Deploy as a github page:`npm run deploy`

Updating the source code can be done using Git bash.

Content update

Pages

Contents on 'About' and 'Guide' are defined in `src/Pages.jsx`. Then, any update on any of those pages can be conducted by modifying the source file.

Images

As stated above, image files are stored in `./public/` directory, including icons for visual summary of the listed evaluations.s

Miscellaneous cases

1. **Changing fields for filtering**. Go to `./src/config.jsx` and modify `columns` object. Object keys represent the short names of the fields. Object values should be consistent to the keys in `./src/data/impact_table.json`.

```
export const columns = {
    'country':'Country',
    'sector':'Sector',
    'type':'Type',
    'funder':'Funder',
    'yearsOfInvestment':'Years of Investment'
    }
```

2. **Changing sector classes**. In the current version, the evaluations are categorised into 7 sectors (from agriculture to WASH). This categorisation can be modified in the future. In addition to the changes on the records stored in `./src/data/impact_table.json`, `iconMapper` object in `./src/config.jsx` should also be modified accordingly. This object maps the sector name to the associated icon that is kept in `./public/`.

```
export const iconMapper = {
    'Agriculture': './icon-agriculture.png',
    'Maternal Health': './icon-maternal.png',
    'NTDs': './icon-ntds.png',
    'Newborn Health': './icon-newborn.png',
    'Nutrition': './icon-nutrition.png',
    'Performance Management': './icon-management.png',
    'WASH': './icon-wash.png'
}
```

3. **Adjusting fields/columns**. As described in Impact table section, there are several keys/fields recorded in the table. More fields can be added to the table. Following this kind of changes, editing form should be adjusted. In `./src/config.jsx`, there is `fields` constant defining the field name/title, type, and default value.

```
export const fields = {
    'evaluationID':{
        'title':'EvaluationID',
        'type':'text',
        'info':'Should be unique number',
        'default':1+Math.max(...evalIDs)},
    'investmentName':{
        'title':'Investment Name',
        'type':'text',
        'info':'',
        'default':''},
    ...
    'funder':{
        'title':'Funder',
        'type':'select',
        'options':['CIFF','Implementing Organisation','CIFF + Implementing
Organisation', 'CIFF + Other Funder'],
        'info':'', 'default':''},
```

```
    'country':{
        'title':'Country',
        'type':'country',
        'options':countryList,
        'info':'Press CTRL to select multiple countries',
        'default':''},
}
```

Here is additional note on the type of input.

type	description
text	text input
select	select from drop-down list
country	multi-select from drop-down list of countries and regions
rte	rich text editor

4. **Adding new evaluations**. Please go to Data structure and update section.

5. **Modifying information to display**. In `./src/config.jsx` there are constants defining information shown on the Info Panel on the right side of the portal and on the Detail Information:

    - `shortContent`: information to display on the country specific list of evaluations.
    - `midContent`: information to show on the right side of the Detail Info panel.
    - `longContent`: more extended information to show below the evaluation-specific map.

    Beside these constants, some fields like `Regions` and `Published results` are specifically defined in `./src/PanelInfo.jsx`.