

The Geospatial Observatory

Final Report – July 2016

Jason Sadler, Homme Zwaagstra, Craig Hutton, **GeoData**

Thanassis Tiropanis, Xin Wang, Ramine Tinati, **ECS**

Project Outline

The Web Observatory provides a data infrastructure allowing users to engage with data sets and analytics on a global distributed scale.

The Geospatial Observatory project explored the introduction of geospatial standards, protocols and visualisation tools to bring ‘geo’ capability to the Web Observatory. The resulting prototype enables users to discover, access, visualise and disseminate such data without barriers for entry, such as a need for web development or GIS mapping expertise, or resorting to local systems or proprietary services in the cloud. Outputs comprise:

- an integrated geospatial service, allowing non-expert users to store spatial data in common formats, and make them available via the WSI Web Observatory infrastructure, using open protocols and standards
- a visualisation toolkit, enabling users from a range of disciplines and IT backgrounds to create and share spatial data and maps directly from within the Web Observatory
- a new GeoAPI and extensions to the existing Web Observatory Application Programming Interfaces (APIs) to enable access of all functionality by data scientists and programmers
- ongoing engagement activities to validate these developments and provide exemplar projects within the web and data science communities as well as a key international development community, leading to opportunities for large scale funding applications.

The project consolidated and extends work from previous WSI stimulus funding for the development of the Hampshire data portal and responds directly to user requirements and “journeys” witnessed during cross-disciplinary Datathon events, which helped established the need, value and potential impact of the development, which can now be utilised in a number of similar forthcoming events organised by WSI and the University.

Activities and Outcomes

Initial work on the project comprised meetings and discussions between the GeoData and ECS development teams to design and specify an architecture for delivering Geospatial functionality and interfaces to the Web Observatory (WO), taking into account the present (Southampton) WO system and user model, and the complexities and challenges of delivering geospatial data directly and through online maps. Constructive brainstorming sessions led to the outline specification for an architecture model and an initial ‘GeoAPI’, which GeoData would develop and present for ECS to integrate into the existing WO. Illustrations of both documents are included in Appendix I.

Development, documentation and testing of the first version of the GeoAPI was completed earlier in the project, with refinements and further developments added following meetings between the teams. A demonstration web template was also developed by the GeoData team, incorporating all of the proposed functionality into a standalone tool (with facility to export selected data sets into an external JSFiddle for further customisation and development), which could be integrated within the extended Web Observatory interface (example screenshots are included in Appendix II).

Technical Architecture

As described in the appended documents, the GeoAPI provides new functionality to enable the WO interfaces (user and API) to register sources of geospatial data, for later discovery, visualisation and external re-use by users, maintaining the system model provided by the WO and providing additional transformation and overlay capabilities for geospatial data sources, using standard open source tools (e.g. GDAL, Leaflet).

Expenditure

£14,750 was expended, fully utilising available budgets, comprising £4500 in ECS time and £10,250 in GeoData time. Many additional *in-kind* days have been contributed to the project, where the technologies have been utilised and further developed on other projects, such as WorldPop, with improvements feeding back into the development, use case and proposal development time (Craig Hutton and Chris Hill) provided in-kind.

Collaboration with external stakeholders

In addition to the regular meetings between GeoData and ECS developers, the activity described above has enabled the teams to explain how the new architecture and functionality will apply to real project 'use cases', or to scenarios similar to the July 2015 Humanitarian Datathon, where this development would facilitate instant discovery and combination of spatial data and creation of shareable online interactive maps. This has enabled Craig Hutton and Chris Hill, focused on the potential use of the Geospatial Web Observatory within new International Development projects, to factor the concepts into existing and forthcoming proposals. The concepts were also presented to delegates at the February 2016 Disaster Management Workshop and are being utilised within other new developments undertaken by ECS WO researchers.

The final development of the fully functional GeoAPI and integration within an extended WO, due to other project deadlines, has occurred towards the end of this project, and the teams are committed to continue working collaboratively to validate the Datathon type use cases and as the basis for further funding applications, particularly within a multi-disciplinary International Development context, specifically building upon a Ganges Migration study, combining environmental and social media data, with international stakeholders from the CARIYA funded DECCMA consortium (Bangladesh, India and Ghana).

Impact Review

The main impact of the project has been the extension of the Southampton Web Observatory with geospatial functionality, enabling data 'producers' to disseminate spatial data without knowledge of web technologies or access to server resources, and users to discover, access, combine, visualise and share those data, which was previously inaccessible to them. The anticipated impact for future Datathon and Hackathon events will be significant. These developments have provided the basis for development of the proposed exemplar projects (devised by Craig Hutton), an International Development case study focussing on Ganges Migration, with international stakeholders from the CARIYA funded DECCMA consortium (Bangladesh, India and Ghana). A potentially large number of remote sensed and socio-economic geospatial data sets will shortly be imported to the Geospatial Web Observatory, relating to Land use and Land cover, assessing environmental impact on assets and Ecosystem Services. This will produce a powerful test use case and demonstrator from the project, and form the basis of a transferable methodology for forthcoming trans-disciplinary funding applications, including the exemplar outlined below.

Further, it will provide the basis for enhanced search features within or across Web Observatories using Geo terms. This can significantly benefit the WO community as researchers will be able to perform longitudinal studies using a variety of data sources related to specific geographical areas.

Future project development

This project enhances the Southampton Web Observatory with basic functionality to visualise geospatial data. To further improve the functionality, the following advanced features have been planned for future work on the project.

- **Searching datasets using geospatial queries**
For geospatial datasets it's a natural requirement to search them using geo terms. By further developing the GeoAPI, geospatial queries can be supported on geospatial datasets. Users can combine keyword based search and geospatial queries to more accurately locate datasets.
- **Retrieving geospatial data from databases (MySQL, MongoDB, SPARQL) and files (CSV)**
Geospatial data can be stored in different types of databases. The WO support several databases including MySQL, MongoDB, SPARQL, and CSV files. Expanding the GeoAPI to support not only geospatial services but all those other databases would significantly enhance the functionality of the Web Observatory.
- **Retrieving geospatial data using the Web Observatory API**
The Web Observatory API enables users to build analytics apps based on datasets listed in the Web Observatory. Enabling the GeoAPI to directly consume raw geospatial data from the Web Observatory API provides a powerful tool to the users to work with geospatial data, and to build geospatial-aware apps.
- **Adding support for real-time data streams**
Many datasets in the Web Observatory are real-time streams. Adding support for real-time streams to the GeoAPI can capture the dynamic aspects of data. It can enable users to visualise real-time events on maps that update as things moving on. It's an invaluable feature that is not currently found in any platform.
- **Geospatial analytics**
Adding support for common analytical operations on geospatial data reduces the barrier to analyse data in geospatial terms. A crucial operation would be grouping data by their geospatial hierarchy to show aggregated result at different granularities. Based on this geospatial "Group By" further operations such as geospatial MAX, MIN, Average can be implemented in a straightforward manner.

Appendix I – Initial Specifications

Outline functionality and application flow from early development team meetings:

* Example use case we want to aim for

- user searches for Hampshire on WO
- find Aerial photography (TMS) and Geotagged Tweets
- look at detail page on WO, see context map and other useful info
- click "open in new map" / "visualise data" on Aerials - opens in new map window
- go back to WO, click on tweets, same process - adds to Aerials in map window

* Registration of georeferenced data

- files/URLs of GeoJSON, KML, Geotagged tweets
- GIS files (Shapefiles, Geodatabases)
- database tables
- use of GDAL to handle all/most of the above (user defined, or template driven VRTs)
- OGC / Tile services
- First step: register a Tile service, e.g. Hampshire Aerial
- WO extended to include basic geospatial metadata (extent, data/plugin type, coord system, etc)
- Second step: add GDAL plugin, test with registration of a simple GeoJSON file

* Querying the observatory and finding geospatial data

- potential for map based or other spatial query (future)
- display context of spatially referenced data (ref map, etc)
- First step: show spatial data type and metadata on results/details pages
- WO data API to include ability to retrieve raw geospatial data (via endpoint)
- Second step: display context map on summary page

* Select & visualise spatially referenced data

- open within an OL or Leaflet map
- First step: open a new window with a single tile service added
- Second step: add another point/vector source to the above
- future - create a JSFiddle type sandbox where they could customise/extend further

* Extend WO API with ability to do the above programmatically

- API for all new features, used by WO to extend interface
 - First step: (basic) data API for providing endpoint from any source of data (often via GDAL)
-

High level GeoAPI specification:

Web Observatory (ECS)	GeoAPIs (GeoData)
<p>Data Set Registration</p> <ul style="list-style-type: none"> is it geospatial? <ul style="list-style-type: none"> No Yes + Type + URI (or data) -> Validate() <ul style="list-style-type: none"> save output in WO database <p>types: TMS (mainly pass through) Shapefile - small - URI is pointer to directory (shared filesystem?)</p> <p>tasks:</p> <ul style="list-style-type: none"> add dropdown and html form entries to registration page call GeoAPI on validate, populate form/db with results 	<p>Validate(type, URI) -> Invalid Valid + Metadata</p> <p>either internal to Validate() or as new ConvertData() api...</p> <ul style="list-style-type: none"> convert via GDAL from input format to suitable storage/output format -> geoJSON KML etc + context image saved locally, used in forthcoming API call
<p>Data Set Search - Result view</p> <ul style="list-style-type: none"> if geospatial, display metadata <ul style="list-style-type: none"> display context map -> GetContextMap() show links for data access (raw, map, qgis, gdal) <ul style="list-style-type: none"> -> GetData() etc <p>tasks:</p> <ul style="list-style-type: none"> add GeoAPI image link to results view requesting context map add various links for data access via GeoAPI calls 	<p>GetContextMap(type, URI, metadata) -> image</p> <p>[potential for caching this image, WO or API side?]</p> <p>GetData(type, URI, metadata) -> URI geoJSON kml etc.?</p> <p>[this method might be overloaded for different data classes/types, e.g. raster, vector, etc]</p> <p>GetQGISProject(type, URI, metadata) -> QGIS Project file</p> <p>GetGDAL(type, URI, metadata) -> GDAL URI</p>
<p>Display Map</p> <ul style="list-style-type: none"> open OpenLayers/Leaflet HTML/CSS/JS template (maybe jsfiddle) <ul style="list-style-type: none"> invoke JS method to add layer: GetData() -> results -> JS <p>tasks:</p> <ul style="list-style-type: none"> embed GeoData provided templates within WO workflow add map display links to open template and invoke JS function to add layer 	

Live GeoAPI testbed interface using Swagger.IO:

swagger Explore

The Geo Web Observatory API

Manage geospatial resources within the Web Observatory: adding datasources; retrieving datasources; generating thumbnails; generating QGIS projects.

Created by GeoData
See more at <http://www.geodata.soton.ac.uk>
[Contact the developer](#)

default [Show/Hide](#) [List Operations](#) [Expand Operations](#)

POST /datasources

Implementation Notes
Create new datasource

Parameters

Parameter	Value	Description	Parameter Type	Data Type
payload	<input type="text" value="(required)"/>		body	Model Schema

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
202	Resource queued for processing		

[Try it out!](#)

GET /datasources/{datasourceID}

Implementation Notes
Retrieve datasource with given id

Response Class (Status 200)

Model Schema

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
datasourceID	<input type="text" value="(required)"/>	Datasource ID	path	Integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
404	Not Found		

[Try it out!](#)

GET /datasources/{datasourceID}/project.qgs

GET /datasources/{datasourceID}/thumbnail.png

GET /queue/{queueID}

[BASE URL]

Appendix II – Application Screenshots

Demonstration web template:

GeoAPIObservatory Widget test

[Earthquake locations](#) [Some worldpop tiles](#)

Map preview

Drag layers to change order

- ☒ earthquake locations
- Color picker
- ☒ some worldpop tiles
- Transparency

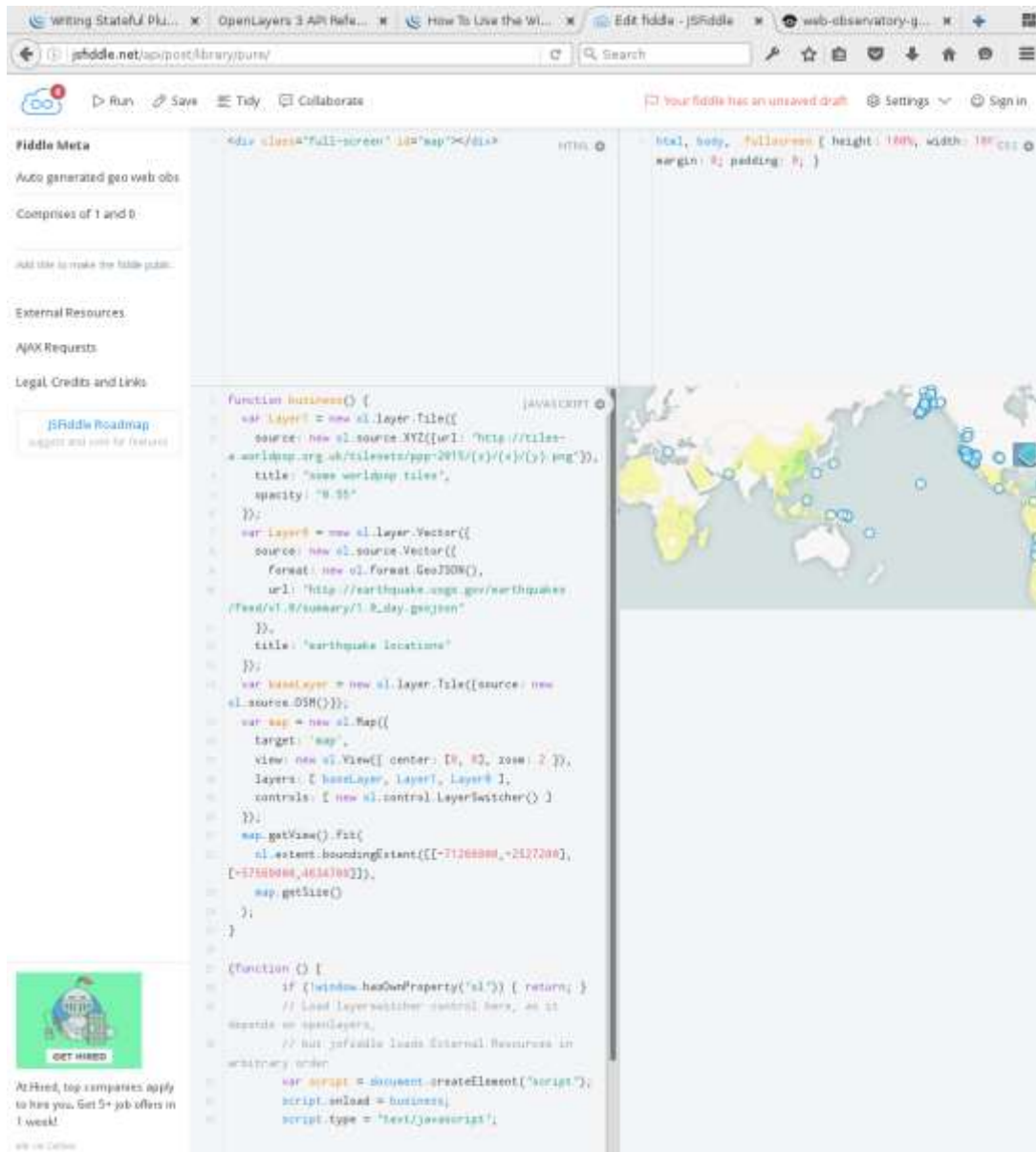
[Generate interactive Fiddle with selected sources](#)

Code preview

```
function business() {
  var Layer1 = new ol.layer.Tile({
    source: new ol.source.XYZ({url: "http://tiles-a.worldpop.org.uk/tilesets/ppp-2015/{z}/{x}/{y}.png"}),
    title: "some worldpop tiles",
    opacity: "0.55"
  });
  var Layer0 = new ol.layer.Vector({
    source: new ol.source.Vector({
      format: new ol.format.GeoJSON(),
      url: "http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/1.0_day.geojson"
    }),
    title: "earthquake locations"
  });
  var baseLayer = new ol.layer.Tile({source: new ol.source.OSM()});
  var map = new ol.Map({
    target: 'map',
    view: new ol.View({ center: [0, 0], zoom: 2 }),
    layers: [ baseLayer, Layer1, Layer0 ],
    controls: [ new ol.control.LayerSwitcher() ]
  });
  map.getView().fit(
    ol.extent.boundingExtent([[ -71266000, -2527200 ], [ -57569000, 4634700 ]]),
    map.getSize()
  );
}

(function () {
  if (!window.hasOwnProperty('ol')) { return; }
  // Load layerswitcher control here, as it depends on openlayers,
  // but jsfiddle loads External Resources in arbitrary order
  var script = document.createElement("script");
  script.onload = business;
  script.type = "text/javascript";
  script.src = "https://cdn.jsdelivr.net/openlayers.layerswitcher/1.1.0/ol3-layerswitcher.js";
  document.getElementsByTagName("head")[0].appendChild(script);
})();
```


Exported to JSFiddle for independent development:



The screenshot shows a JSFiddle web application. The browser tabs at the top include 'Writing Stateful P...', 'OpenLayers 3 API Ref...', 'How To Use the W...', 'Edit fiddle - JSFiddle', and 'web-observatory-g...'. The address bar shows 'jsfiddle.net/api/post/library/pure/'. The JSFiddle interface includes a left sidebar with 'Fiddle Meta', 'External Resources', 'AJAX Requests', and 'Legal Credits and Links'. The main area is split into three panes: HTML, CSS, and JavaScript. The HTML pane shows a basic map container. The CSS pane shows a full-screen map style. The JavaScript pane contains the logic for loading map tiles and earthquake data. The map itself is displayed on the right side of the JavaScript pane, showing a world map with blue dots indicating earthquake locations.

HTML

```
<div class="full-screen" id="map"></div>
```

CSS

```
html, body, full-screen { height: 100%; width: 100%; margin: 0; padding: 0; }
```

JAVASCRIPT

```
function business() {
  var layer1 = new ol.layer.Tile({
    source: new ol.source.XYZ({url: 'http://tiles-
    a.worldmap.org.uk/tiles/xyz/256/{x}/{y}.png'}),
    title: 'www.worldmap tiles',
    opacity: '0.55'
  });
  var layer2 = new ol.layer.Vector({
    source: new ol.source.Vector({
      format: new ol.format.GeoJSON(),
      url: 'http://earthquake.usgs.gov/earthquakes/
      feed/v1.0/summary/1.0_day.geojson'
    }),
    title: 'earthquake locations'
  });
  var baseLayer = new ol.layer.Tile({source: new
  ol.source.OSM()});
  var map = new ol.Map({
    target: 'map',
    view: new ol.View({ center: [0, 0], zoom: 2 }),
    layers: [baseLayer, layer1, layer2],
    controls: [ new ol.control.LayerSwitcher() ]
  });
  map.getView().Fit(
    ol.extent.boundingExtent([[71268888,-2127288],
    [-17588888,4634782]]),
    map.getSize()
  );
}

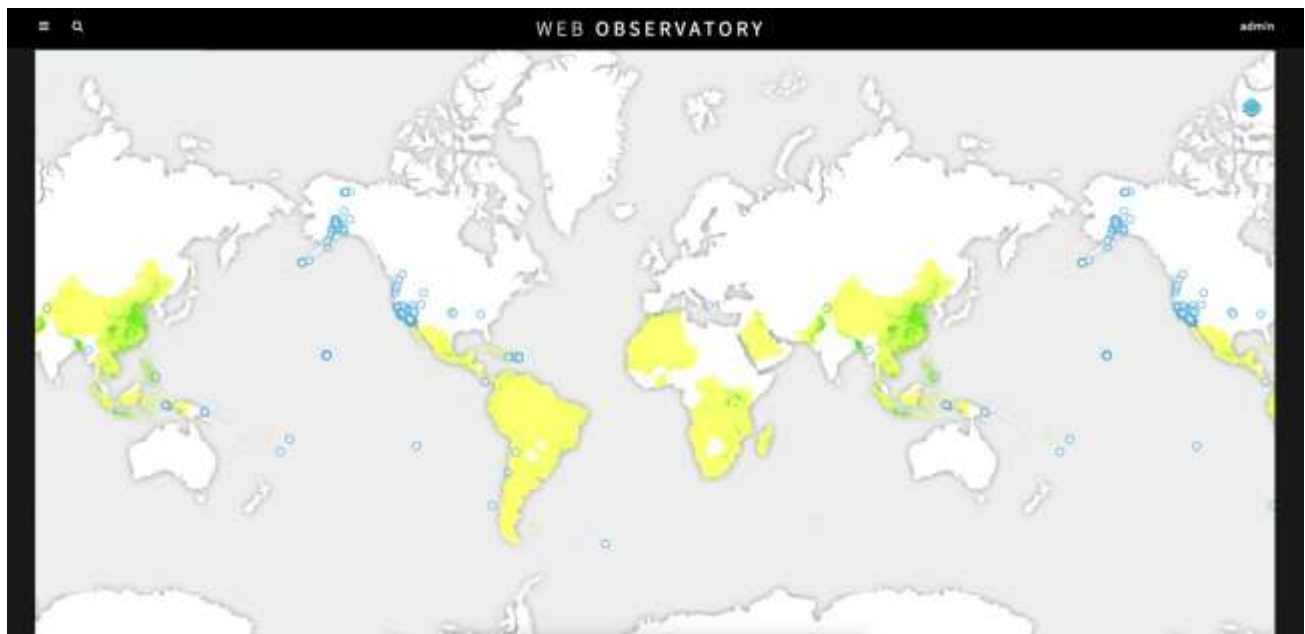
(function () {
  if (!window.hasOwnProperty('ol')) { return; }
  // Load LayerSwitcher control here, as it
  depends on openlayers,
  // but jsfiddle loads External Resources in
  arbitrary order
  var script = document.createElement('script');
  script.onload = business;
  script.type = 'text/javascript';
}
```

GET HIRED

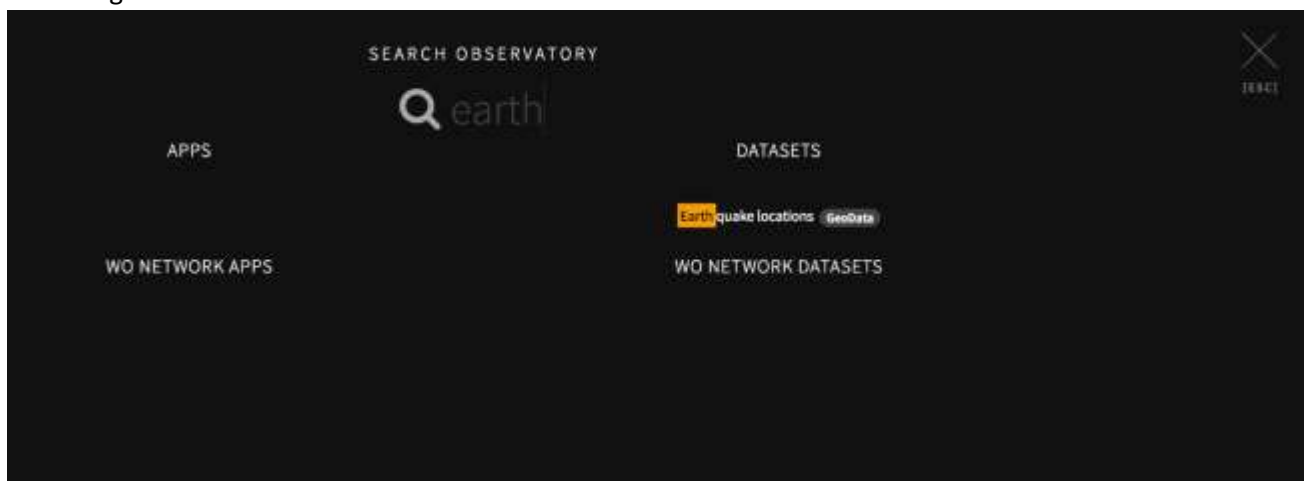
At Hired, top companies apply to hire you. Get 5+ job offers in 1 week!

[Get Hired](#)

Incorporated within the Web Observatory:



Searching for Geo datasets:



Metadata of a Geo dataset: