figshare for institutions

A closer look at...

figshare's API

Jez Cope
Research Data Manager at the University of Sheffield

Marcelo Paulino
Software Developer at the University of Sheffield

Casey Gibbs
Digital Services Manager at St. Edward's University

Key Points

- figshare has an API (application programming interface) which can be used to ingest or harvest data automatically from figshare for batch uploading, reporting, visualizing, or displaying data in different applications and programs. More information on our API can be found at docs.figshare.com
- There are some features that exist via the API that don't exist in the interface, so it's worth exploring how you can use it.

How have you used figshare's API?

Casey Gibbs: So far, we've done two things with the API. First, we connected it to a web endpoint that we created to represent our institutional repository (IR) at the time, ir.stedwards.edu, that lists, among other things, all of our content in figshare. That's just a basic API call. Second, we have a weather balloon project (ir.stedwards.edu/natural-sciences/ozone) where we route data into figshare items and have a frontend for showing that data. This uses the D3.js library to visualize some of the data in those items.

This all began around the time when I joined St. Edward's and we were trying to get our IR off the ground. We had a new Dean of Natural Sciences who had just started at the university having moved from a different university where for several years he had directed a project collecting weather balloon data. He had several different launch sites around the world and several times a year per launch site, he would send up the weather balloon to collect data in a raw format. He had proprietary software that processed that raw data in its binary format, produced a structured text format of the data, and fed that structured text to different proprietary software that produced a PDF, TIFF, or JPEG visualization of that data. He had a website that he published this data to and he used Dreamweaver to manage it. This site would include the links to the data he harvested.

He came to us and wanted us to reproduce that at St. Edward's. However, we wanted to remove the manual workflow and instead automate the deposits of his data into a system and automate publication of that into a web endpoint. Having just signed on with figshare and having just set up our own web interface for revealing what we have in figshare, we thought we could create a web endpoint just for this project. We would automate depositing his visualizations into figshare and then have a web endpoint that would query figshare's API for an item and generate that table of links automatically.

So, we started building that. In the process, we realized a few things. Just uploading his data to figshare gave us an architectural problem. We weren't sure how to represent his data as figshare items. Should we compile all of his launches since the beginning of time for that launch site into a single figshare item or should we have an arbitrary subdivision?

In the process of deciding how to organize the data, we realized it would be better to use D3 to render on-screen, dynamic visualizations of that data rather than uploading his PDF visualizations of these launches. That way, someone can at least get a sense of whether they want to download the PDF.

That lead us to where we are now. We now have a single figshare item for each launch site that compiles all of the data since the beginning of time and uses specified file names for each launch file that includes a timestamp of that launch. When you open up the page for that site, it lists figshare items which correspond to launch sites. You click on a figshare item corresponding to a launch site and it looks at the list of all the files in that figshare item and takes out the ones that represent the structured text files which include the timestamp of that launch and render just those as links. Page two is a list of links for dates of launches at that site. If you click on one of those, it first extracts from that text file at the top of the page all of the basic metadata for that launch: the PI, the date, the launch site, climatological data, etc.

While that's rendering on screen, in the background, we're also querying figshare again to list every other file from that item with the same timestamp. It then renders links in a sidebar so you can pull all his original PDFs and visualizations in a sidebar. While that's happening, we're also sending the data in that text file to D3.js to render his visualizations dynamically. Also, for almost all launches, he has a KML file, which is a specification for geographic data. We're also taking that KML file and sending that to Google Maps' API to render a nice, interactive map at the top of the display. So, we're still providing access to all of his original files but nothing manual happens.

Jez Cope: We're starting to look at visualizations, as well. We were inspired by the work St. Edward's has done as well as the work Martin Hadley has done at Oxford (see his GitHub for more information: github.com/martinjhnhadley). They have put together a showcase of interactive visualizations pulling live data where possible. They picked figshare for the backend because they wanted something that did versioning DOIs properly as this made his implementation a lot easier.

To see Martin's backend data, visit doi.org/10.6084/m9.figshare.3761562.v529

Those visualizations could be hosted anywhere but mostly are hosted on shinyapps.io. They had a training and skills development aspect to this as well so they decided to focus on a single language and went with R because it's popular, growing, and Martin was familiar with it. Having built ORDA as a frontend onto what we had on figshare, it made sense that if we were going to do something similar, to build that into ORDA as well.

ORDA, Online Research Data - orda.shef.ac.uk - is University of Sheffield's data repository. They built this layer on top of their Figshare instance using Figshare's API. To learn more about that, visit: doi.org/10.6084/m9.figshare.5002433.v1

We've been experimenting recently with how to do that and how to get all the metadata for each visualization to be stored within figshare so we don't have to set up a database somewhere else to drive it. We had various conversations and decided to put a machine readable metadata file within the figshare record for the things we wanted to visualize.

Marcelo Paulino: We are now investigating libraries and language like R or Python to help us in this process. Currently, I'm learning about Vega-Lite which is a library that allows us to generate visualizations only using JSON files. The main idea is to build a showcase similar to Oxford's. We aren't pretending to reinvent the wheel.

Casey: It sounds like we had an advantage over you in that our project already supplies its own structured metadata – for every launch, we have that file. That was the inspiration for us from the beginning. For subsequent projects that we've explored, we had to ask ourselves the same questions. Where do we want to put metadata? In the figshare file or in a local database? I don't think there's an easy solution to that, but I like the idea of putting the file into each figshare item along with the data.

Jez: The idea is that anything you can embed in an iframe, we can include. We have a page that includes the metadata and the visualization is displayed directly on that page. You can link through to the raw data hosted on figshare.

Are you trying to promote data visualization as a service to your faculty?

Jez: The ultimate goal is to get more data into our repository. One of the approaches we're trying to take to that is to add value for the researcher. If we can make it easy for people to present their own data in a more compelling way, hopefully they're more likely to deposit their data in the first place. We may have to do some consultancy work to prime the pump, but the main idea is to provide training, documentation, guidance, and templates to encourage people to do this for themselves and make this as much of a self-service as possible.

Vega and Vega-Lite are quite an interesting possibility because there's no programming required – you're just specifying static metadata. The Vega embed library takes the data and renders it using D3.

We're increasingly finding that researchers are starting to pick up bits of R and Python. Shiny makes it very easy to make a very simple plot and turn it into a web application.

Did you know you can publish your data straight from R? Find out more at: github.com/ropensci/rfigshare

Casey: It's a very different situation in the States right now. There's very little public attention paid to how you deal with your data, how you publish your data, and why you would want to publish your data. There certainly isn't the kind of momentum that you have in the United Kingdom. Because of that, we're much more focused on the consultancy model for driving adoption. We're pretty enthusiastic about it, too. We're really excited about putting together a portfolio of services, the biggest of which is education. We want to tell the whole story from application for grant funding to collection of data with the endpoint already in mind to publishing and visualization and so on.

What are you working on next?

Casey: As far as the API is concerned, we have couple of projects coming up. API is collecting oral histories that we will put in figshare and we'll create a custom local endpoint for both collecting new oral histories and accessing them.

Also, for our own internal processes, inspired by our issues with moving files around for figshare with the weather balloon project, we're also working on a command-line client using Elixir so that it will be concurrent to be used for real-time data. We would like to have an open source beta version of that available by Christmas.

Jez: We're just getting ready to launch a minimal viable product for our visualization showcase and will follow that up with more information gathering work with users to get a better idea of where we should go next. We're at a junction in the road as to where we can take this project moving forward.

Do you have any tips for other institutions who are looking at using the API to develop platforms and services themselves?

Marcelo: The API was so useful for us. The ORDA project was done using 100% of the API. I recommend that others look at our portal and try to have some ideas as to how they could do something similar.

Jez: Ask questions. We found early on that there were going to be things we wanted the API to do that it couldn't do or that we weren't sure how to do. So we got stuck in and started asking questions on the Slack forum which was useful.

Casey: For anyone who is starting from scratch learning how to use APIs and how to write code that uses APIs, I would suggest the Jupyter Notebook format. This will get you on the right foot because it documents itself as you go and it can be passed around from one researcher to another. I have previously sent IPython Notebooks to Figshare directly to ask where I'm going wrong with certain code which has been immensely valuable. There are also IPython Notebooks available in the API documentation.

I would also note that there is nothing you can do in figshare's interface that you can't do via the API. There are even some features that exist in the API that don't exist in the web client. It's an extremely powerful tool and an immense part of the value of a subscription to figshare.

Visit sheffield.figshare.com

Visit stedwards.figshare.com