XCITE HYSPLIT and IDEA Apps: How they work

William May

July 15, 2019

Contents

App organization

There are a lot of moving pieces that make up the XCITE HYSPLIT viewer apps – probably more than necessary.

Organizing the data

A python script downloads the data from headnode7. Another python script reads the data files and loads them into postgres– the trajectories and concentrations as geojson and the gridded model outputs and satellite swaths as PostGIS rasters.

Tools Used:

- python
- GDAL
- postgres
- postgis

Code location:

- `pireds:/home/xcite/hysplit`
- `pireds:/home/will/idea`

Serving the data

Flask (python) servers connect to postgres and provide the data as json, png, and geotiff.

Tools Used:

- python
- flask

Code location:

- `pireds:/home/xcite/hysplit/hysplit.py`
- `pireds:/home/will/idea/server.py`

Displaying the data

Like most websites, the app pages are composed of HTML, CSS, and javascript. Javascript functions download data from the flask servers and map it using leaflet. The code leans heavily on the leaflet.TimeDimension extension for mapping time series. Many parts of the interface are created with jquery. The javascript code needed to organize all of this is relatively complex and unwieldy.

Tools Used:

- HTML
- CSS
- javascript
- jquery
- leaflet
- leaflet.TimeDimension

Code location:

```
pireds:/home/xcite/public_html
```

Recommendations

Shiny

Shiny is an R package for creating web dashboards. Shiny automatically generates the required html, css, and javascript, so that dashboard creators can make web apps without needing traditional web development skills. The leaflet package for R can create interactive leaflet maps and integrates with Shiny. Another R package, leaflet.extras, provides an R interface to some leaflet extensions, and work has been done to include leaflet.TimeDimension, though that hasn't been completed (see the github issue here).[1]

To switch to Shiny, some work would need to be done to get leaflet.TimeDimension to work from R. After that the code would be much simpler– the python/flask servers could be abandoned altogether, because Shiny is its own server and R can connect directly to the postgres database. Javascript code for maps, legends, and user interface controls are generated by simple R code, so that the dashboard is much easier to organize.

Required skills

For the Shiny option, most work could be done with some knowledge of R, though experience with web development would be useful for customizations.

For javascript without Shiny, relatively strong javascript skills would be required. Some knowledge of python would be needed for organizing the flask servers.

If new types of data are going to be added, organizing that would require familiarity with SQL databases, and GIS skills are also useful for working with the spatial data formats. Currently the data is loaded into postgres by python, but that could also be done in R or another language.

[1] I'd be interested in working on this if it can be arranged somehow.