Scripts for GCBM-FLINT Course Videos

9th Slide

As I mentioned before it makes the FLINT very scalable, but it does make certain problems very difficult for it to handle.

We cannot deal with awareness of neighboring pixels so we cannot work with fire spread, and we do not really know the exact location of rule-based disturbances.

10

This is a very high-level overview of what the event-driven system looks like. Basically the current pixel data is run through the spatial layers in the database and then it goes to the event sequencer and that's what is firing those life cycle events things like timing init, timing step start, timing step end and the output.

And then all the science models are subscribed.

11

One of the examples of our modules is the growth module. It subscribes to three of those system events, the simulation start- loads the root biomass equation, each time a pixel is loaded, it loads dead organic matter turnover rates and the timings step is going to load or process the current growth curve.

12

CBM science is implemented as modules running on the front platform. The key features are that ecosystem components biomass soil greenhouse gases are represented as carbon pools. There is a yield curve-based growth module, and the disturbances and annual processes are represented as transfers between these carbon pools.

The core module includes   
- biomass growth and mortality  
- dead organic matter and soil dynamics and   
- disturbance impacts - management activities like harvest, natural disturbances like fire and land use changes like deforestation, afforestation.

13

We went through a pretty lengthy process when we developed this new model and the results are nearly identical to CBM CFS 3.

Let’s look at the main differences from CBM CFS3. forest in inventory and disturbance data come from spatial layers instead of database tables.

The location of all the disturbance events is explicit in spatial layers instead of using rules.

It gives out special as well as tabular output for pools and fluxes also it is easier to extend with new models and easier to simulate large landscapes such as the ones that have a huge number of pixels.

14

Here is an example of one of the outputs that we generated using the model. The simulation does not produce this directly but there is a set of advanced python scripts that we used to produce these outputs. They are also freely available along with the ones that generate animations using the spatial output.

15

Let's have a look at the actual structure of a GCBM project. We start off with some spatial data like disturbances and forest inventory data, along with a database of non-spatial modelling parameters like growth curves and disturbance m. This passes through a set of python based pre-processing tools to output data in GCBM readable format.

And the along with GCBM configuration files in json format go for GCBM simulation. It produces a set of spatial outputs as well a SQL database output.

16

Talking about the data that we need for running GCBM. On the spatial side, we need At least a map of the initial forest age or the time since last disturbance and the classifiers. It is recommended to have a mean annual temperature layer which affects the decay rates and reference parameters used are almost similar to that of CBM-CFS3. An administrative, ecological boundary is also used.

In terms of non-spatial or tabular data, it requires a CBM-CFS3 archive index database which contains the library of non-spatial modelling parameters and a CBM-CFS3 styled yield table in CSV or Excel format. Using a country specific CBM-CFS3 archive index database, that is customising it according to using the modelling parameters that are more suited to your geographical region improves the results.

17

Let's look at the archive index database. It contains the non spatial parameters and spatially references parameters  like the decay rates, disturbance matrices, and root biomass coefficients.

It is used in both CBM CFS3 and GCBM.

18

A little more detail on the disturbance events format for GCBM.

We can take pretty much any spatial layer format but usually  on the raster side it'll be a file per year where pixel value equals to the disturbance type or file per disturbance type and year where pixel value equals to 0 or 1, bits indicating disturbed or not disturbed.

We can also take in vector layers and have any number of those layers that contain polygons of completely disturbed areas. What you see on your simulation map is what ends up getting simulated. You will need a couple of attributes in the vector layers, like what's the disturbance year or what's the type of disturbance, etc.

And these will be paired with a lookup table that maps non-standard disturbance type names in the spatial layer to actual names of disturbance types that are understood by the model.

19

Getting into the structure of this training project, the standalone project is a collection of individual pre and post processing tools organised into a workflow to get people up and running with GCBM.

It does not do any sort of fancy pre-processing and is really what you see is what you get. It ingests SPATIAL layers and SIT like data and runs exactly what has been specified by the input data.

20

It is designed to be portable so you can copy it anywhere and run with correct python or GDAL environment

21

It is best suited for the project where the data already exists in the correct format and does not require any pre-processing before-hand.

I just not designed only for this training. All of our real-world projects basically use the same project structure and scripts.

The standalone template is the recommended way to get new projects up and running.

We do some advanced scripting internally, but this is the basic project structure used.

22.

This is the overview of standalone projects workflow. We start off with tiler that converts our raw data into GCBM readable format.

The next is a tool called recliner to GCBM that is responsible for creating the SQL database.

Once we've got the input data in place, we run the actual GCBM model itself. Once the run is finished, we get a couple of post processing tools. The first one is compiled GCBM spatial output, which stitches all the chunks of output to geotiff files. Then there's another tool- compile GCBM results that generates ecosystem indicators in to SQL like database

23.

Let's talk about the outputs given by our standalone project. It's usually a SQL like database, tabular form output that uses SQLite by default and postgres as well.

The spatial output is in envi or geotiff format. After post processing, it gets converted to the standard geotiff format.

And then it produces some flattened reporting tables that are more user friendly. They can be either SQLite or postgres databases.

24

There's a spreadsheet available describing exactly what goes into all those reporting tables. The reporting table format integrates easily with tools like PowerBI desktop, and we can create PowerBI templates to visualise results from different simulations. It has all the indicator names here on the left

25

This is a screenshot of one of the PowerBI templates to visualise the results.

26

The directory structure of the project template  looks like this.

Documentation folder contains some important materials, for example python snippets for configuring the tiler script, and basic assorted documentation.

Then we've GCBM folder That contains the simulation working directory

the next one is input Database that contains the a spatial input data for example archive index database , yield table and all

 next comes the layers which has the spatial layers inside it

Next we have a log files folder, processed output folder, and the tools that contain all the pre-and post processing tools GCBM itself and supporting software.

Then we have the license agreement for using GCBM, it is freely available to everyone.

Then we have the readme and batch file.

27- Project-Rectangle World

Let's have a look at our sample standalone project named as rectangle world.

It has been set up with some exemplar simulation data. the initial python or GDAL environment should be set up for running this.

The spatial layers can be found in the layers folder in the raw subfolder and the yield table is stored in the input database directory as yield dot csv.

28

Let's start by having a look at the input database that we are going to use in this project. This is the inventory shape file.

This can be found in the standalone template folder in the layers subfolder as inventory dot SHP.

There are a couple of key pieces of information  that we need to add in the inventory layer.

Age 2010 is the forest age at the beginning of the simulation.

There are two classifiers for this.

In this project we will call them classifier one and classifier 2 but they can be named anything.

They link our  spatial data to the non-spatial yield table that describes how the forest grows.

29

The next layer in the sample data set is the disturbance layer.

Just like an inventory layer, you can find it in the standalone template folder, in layers subdirectory as disturbances dot SHP.

We have the first attribute as disturbance type which is already understandable looking at the values and then we have the year attribute.

30

Next we have got the non spatial data. This is what our yield table looks like.  This format is pretty similar to CBM CFS3.

This file can be found in the input database folder as yield dot csv.

31  
Following the process for running a project in GCBM, the first thing that will do is configuring and running tiler to process the input spatial data into the GCBM format.

We will run the recliner to GCBM tool to create the GCBM input database.

There is a python script that will configure the GCBM for us.

Next we'll run the model.

And lastly we will run the post processing tools to generate the final output.

Familiar to users of CBM-CFS3, we generate all the usual ecosystem indicators like NPP in database tables.

And then we will convert the raw spatial output to final layers.

32

This slide gives us a basic technical background that we might need to understand some aspects of using GCBM.

Python is a general-purpose scripting language that we use for most of the pre and post processing work.

We will need a basic knowledge of python to run the GCBM.

There is a cheat sheet available in the documentation folder as tiler cheat sheet dot Txt that provides python snippets for the most common tiler use cases

The configuration files for GCBM are in JSON format that stands for JavaScript object notation

It is a common text file format that is used by many software packages.

GCBM and its supporting tools use JSON as a configuration file format.

33

Here are some prerequisites to run GCBM.

You can find the instructions in the readme dot Txt file.

if you have python 3.7 already installed on your machine, simply locate it by existing python installation.

Open a command prompt in tools slash python 3 installer and then type install modules only dot BAT.

In case python 3.7 is not already installed, open a command prompt in tools / python3 installer folder and type install python followed by the path of python.

Python will be installed into the specified path or into a directory called python 3 7 in the c drive by default.

34

since the archive index database is in MS access format so will need to install the access database engine.

and then we have some visual C++ redistributable packages and dot net 4.8 installers.

35

Once all of that is installed you can check that everything is running correctly just by double-clicking the run all batch file.

If everything goes well it's great but if not, here are some troubleshooting steps.

If the tiler script fails it's a python or GDAL issue. We will need to check the environment variables again. You can recheck that all the paths have been set properly and correctly.

If the recliner to GCBM fails its usual access database driver issue.

So you can try switching from the 64 bit version to the 32 bit one by changing the platform variable in the run all batch file. I'll show you the batch file here 😁

At the top there are some configuration lines we have platform variable which refers to access database. We can try changing its value here and hit the save button.

If the model fails to run and it's usually because one of our earlier tests failed.

if the post-processing script fails again it's usually one of those areas that we missed before, either because GCBM failed or a python or access database driver issue.

⏱So if everything runs properly you should see this on your screen when you run that run all batch file.

⏱⏱⏱⏱

We got past running the pre processing tools pretty early and this is actual GCBM running.