"Spatialized Phylogenetic Climate-driven Ecodiversity Simulator" (SpyCiEs) v.0.0

Alexandre Pohl
BiogÃľosciences, UMR 6282, UBFC/CNRS,
UniversitÃľ Bourgogne Franche-ComtÃľ,
6 boulevard Gabriel, F-21000 Dijon,
France

Last updated: June 14, 2021

Contents

1 Rationale

SpyCiES is a biodiversity model built upon the model of Brayard et al. (2004, 2005), translated to `python` in 2021 by A. Pohl, in the goal to add new developments to the model code.

You are reading the support documentation, the full and most uptodate version of which can be found at (and cloned from) that address:

```
git clone https://github.com/alexpohl/biodiv_model_doc
```

It is expected that the readers has basic knowledge of linux commands and has been granted access to a computer or cluster.

2 Differences with the historical model

Here are the main differences with the model of Brayard et al. (2004, 2005):

1. Speciation and extinction rates are very different, notably due to the much lower horizontal model resolution (36×36, compared to 120×120).

2. Since Brayard et al. (2004, 2005) showed that currents to do impact the final result, and considering that we're mainly interested in equilibria, we got rid of currents.

3. The model does not converge toward a stable dynamic equilibrium, possibly due to the longer integration time or inadapted speciation and extinction rates... The model is thus free to drift and there is no capping of local biodiversity (although the code still has the provision to add it, for backward compatibility).
4. The model is now coded in python rather than IDL.

3 Obtaining the code

The model code is hosted on **Github**. You can visit the webpage and download an archive of the code, but the best solution to obtain a full copy of the model code with possibility to easily update it later, is to **clone** the repository:

```
git clone https://github.com/alexpohl/biodiv_model.git
```

Remark: You may get an `ERROR 404 - Page not found` when trying to open the GitHub link above while not logged in.

The model will be executed in that directory; you should thus clone (download) the directory at a location where you can execute programs and write output files. Typically, on the CCUB cluster, this is gonna be on your `WORKDIR` (i.e., not your `HOME` nor your `ARCHIVE`).

In any case, the code is **password-protected** and preliminary steps will consist in:

1. Creating a GitHub account if you don't have one.
2. Getting access to the model repository (on request to A. Pohl).

4 Running the model

Requirements SPyCieS has been developed to run on **Linux** (clusters). It has not been tested on other operating systems, although it is coded in python and should be fully usable on MacOS and Windows. You need a **Python 3** install and may need to install new packages (probably using `pip install --user package_name` [replace `package_name` with the name of the package you wanna install, e.g., `rasterio`]).

Modules On most linux clusters, many programs (such as Python 3) are already installed; all you'll have to do is loading the right modules (in the case of Python, the packages coming by default with the module may not be sufficient and you may still have you install additional ones, see above). Use `module avail` to list available modules, `module purge` to unload every module, and `load module_name` to load a specific module. You can include such module management code into your `.bashrc`, `.czhrc` or `.cshrc` file (pick the right one according to your shell type, determined using `echo $shell`) in your `$HOME` directory. For the record, here are the modules that I suggest loading on CCUB:

```
 Currently Loaded Modulefiles:
1) intel/2020
2) python/3.6/intel/2019
3) nco/4.7.7/intel/2018
4) cdo/1.9.2/intel/2018
5) gcc/9.1.0
6) R/3.6.2/gcc/9.1.0
```

Running the model interactively The **main program** can be found in `mainprog.py`. It also uses miscellaneous python functions gathered in the `source` directory. `Mainprog.py` requires one positional argument, which is the **userconfig**. Here is the line you should run into your linux terminal (from the model root directory) to execute a model simulation interactively (executing this precise line will run the model example experiment, for which all required input files are provided with the model code):

```
python mainprog.py userconfigs/userconfig.py
```

This will create an output directory with same name as the `userconfig` file used, in the `output` directory, or alternatively at any location provided using parameter `par_pathout` in the `userconfig` file. Please note that the content of the output directory will automatically be deleted when a new simulation of same name is launched – so be careful. You'll see key numbers displayed interactively on your screen as long as the model runs. You can interrupt the simulation at any time by simultaneously hitting keyboard keys `CTRL` [control] + `C` [letter C; no capital: do not hold shift at the same time].

The interactive mode is convenient to rapidly test your code but its use is very limited: the simulation stops when you close your terminal window or log out, and it does not permit running several simulations in parallel. In order to overcome those limitations, your are highly encouraged to submit simulations in batch mode.

Running the model in batch mode A small utilitary is provided to directly submit a batch Job. It has been designed to work on the regional cluster (CCUB) but can be easily adapted to other clusters. The basic usage consists in typing:

```
python runbatch.py userconfigs/userconfig.py
```

You can also do the same but also assign a name to your simulation / batch job:

```
python runbatch.py userconfigs/userconfig.py jobname
```

After submitting batch jobs, you'll be able to get a list of your currently running simulations using `qstat`. If you want to stop a model run, type in `qdel job-ID`, `job-ID` being the unique, 7-digit simulation ID provided in the first column of the output of the `qstat` command. For more information on the CCUB cluster, please read `https://ccub.u-bourgogne.fr/dnum-ccub/spip.php?article959`.

The output log of batch jobs is redirected to `./logs`; it is useless most of the time but can sometimes be used to keep track of execution issues and the model execution time.

5 Analyzing the model output

Output files The model regularly outputs **snapshots** during its integration (every `par_nc_saving_freq` timestep(s)) and keeps track of every timestep in the log file. Once the run finished, additional files are generated:

1. **timeseries.nc**: this file contains time series, with data for each single time step.
2. **history_timesteps_#to#.nc**: this file gathers all saved snapshots into one single output file.
3. **final_cologrd.nc**: this file contains, for each species present at the end of the model integration time (but not for species that disappeared), the final spatial extent together with the temperature range defining the thermal niche of the species (i.e., minimum and maximum temperatures).

Should the simulation not reach its final timestep, `history_timesteps_#to#.nc` would not be output. In that case, a python routine is provided to assemble the missing timeseries: `utils/evol.py`.

Plotting output data Most output data comes in the from of self-describing **NetCDF** files, which can be plotted in various ways: using Matlab, Python or Ferret but also GIS softwares, Panoply... I personally suggest using Ferret, because it's very easy to use to rapidly explore data. Ferret is not installed on the CCUB cluster and will have to be installed by downloading the github distribution in the form of a `tar.gz` file and following the installation procedure, see `https://ferret.pmel.noaa.gov/Ferret/downloads/ferret-installation-and-update-guide`. A user guide can be accessed at `https://ferret.pmel.noaa.gov/Ferret/documentation/users-guide`. For instance, use the following lines to plot any snapshot output file:

```
ferret # will start ferret
use snapshot_timestep1999.nc # will load the output file
show data # will show the content of the file
plot BIODIV[X=@ave] # will plot the latitudinal diversity gradient, i.e., zonal average of
biodiversity
```

As you can see, Ferret is really concise! It also permits assembling such command lines into a script with extension `.jnl`, that you'll call and run inside Ferret using `go myscript.jnl`.

A basic python script is also provided (`./utils/plot_LDG.py`) that plots, for any given number of output paths, the latitudinal diversity gradient normalized to maximum diversity (mean, and enveloppe corresponding to $\pm 2\sigma$, calculated from a given timestep to the end of the model run). The normalization permits comparing the results of different experiments that may have run for variable durations or taken different trajectories, and whould thus be characterized by very different total numbers of co-existing species.

6 FAQ

6.1 HOW TO: check that the model reached equilibrium

It is important to check that the model reached equilibrium before interpreting the results. In short, in this case, it consists in determining if the simulated latitudinal diversity gradient reached a final shape, or if it is still significantly evolving.

Because the number of species keeps growing during the model integration time, absolute values of number of species per latitudinal band will necessarily keep changing. In order the really have a look at the **shape** of the latitudinal diversity gradient, it is useful to **normalize** local diversity by the total number of species. Then, it is possible to determine if the normalized latitudinal diversity gradient still evolves, or not. An interesting tool is the python script `./utils/plot_LDG.py`, which gives the mean latitudinal gradient over the last # model timesteps, and also the enveloppe (2σ, thus representing the 95% confidence interval).

In case equilibrium hasn't been reached at the end of the model run, you may want to run a longer simulation by increasing the total number of timesteps: parameter `par_n_timesteps` in the `userconfig`. Unfortunately, depending on the initial seeding point and continental configuration (among other things), the number of timesteps required to reach equilibrium will vary and the model does not include any automatic convergence criterion so far. We tried capping total biodiversity using random global extinctions (by setting `opt_cap_max_global_diversity = True`) but this workaround appeared to significantly destabilize the simulated latitudinal diversity gradient and was therefore abandoned.

6.2 HOW TO: play with the model parameters

Many model parameters are included in the `userconfig` file. In order to determine the impact of a given model parameter on the results, one traditionally conduct a **sensitivity analysis** to this parameter by repeating several times the exact same simulation, except that the parameter value is changed (all other things being kept equal). The main model parameters to change are the following:

1. **par_dataset**: the sea-surface temperature field. A flatter latitudinal temperature gradient, supposedly typical of warmer climatic periods of the geological past, would modify the simulated latitudinal biodiversity gradient.

2. **par_imposed_seeding_pt_indices**: the original seeding point. An important question is to determine if the original seeding point impacts the final shape of the latitudinal diversity gradient, or not.

3. **par_sst_distribution**: the shape and width of the species tolerange range. Two options are coded: `fixed` considers that all species have a given, identical, thermal tolerance, while `gaussian` extracts the thermal tolerance from a gaussian distribution of parameters μ and σ. A small python script

(`./utils/plot_distributions.py`) permits visualizing the gaussian distribution associated with given values of μ and σ.

6.3 HOW TO: implement a new source of input data

If the source of data has not been previously implemented, you'll have to add it as another `elif` statement in `./source/fun_read_gcm_output.py`. Then, you can define the dataset info in `./input/define_datasets.py`.

References

Brayard, A., Escarguel, G., and Bucher, H. (2005). Latitudinal gradient of taxonomic richness: Combined outcome of temperature and geographic mid-domains effects? *Journal of Zoological Systematics and Evolutionary Research*, 43(3):178–188.

Brayard, A., Héran, M. A., Costeur, L., and Escarguel, G. (2004). Triassic and cenozoic palaeobiogeography: Two case studies in quantitative modelling using IDL®. *Palaeontologia Electronica*, 7(2).