GEODIAG

Geoscientifc Diagnostics Open Platform version 1.0.0-rc1

Li Dong

Table of Contents

1 Introduction

GEODIAG utilizes NCL as the base for constructing a diagnostics platform.

2 Basic usage

2.1 Installation

GEODIAG is version controlled by using `git` and hosted at `GitHub`. It is recommendated to obtain GEODIAG through `git` as

```
git clone https://github.com/lasg-model/geodiag
```

In this way, the local copy can be updated easily by

```
geodiag update
```

After downloading GEODIAG to `<geodiag-root>`, modify the configuration of BASH (e.g. `.bashrc`) as

```
source <geodiag-root>/setup.sh
```

Then login again, you should be able to invoke the command `geodiag`.

2.2 Library usage

```
load "$GEODIAG_ROOT/geodiag.ncl"
```

2.2.1 China Map

The Chinese border has long been subject to the harassment of some other countries, even in scientific field (see Figure 2.1). So GEODIAG provides an utility to plot correct border. This utility tries to minimize its trace. Two functions are mandatory (`setup_china_map` and `attach_china_map`), which clip the user plotting codes.

```
res = True
res@isShowProvince = False
res@isShowSouthChinaSea = True
res@isAddMask = True
res@isShowRivers = True
res@riverColor = "black"
res@riverThickness = 0.5
res@boundaryColor = "black"
res@boundaryThickness = 0.5
setup_china_map(res)
... ; user plotting resources
plot = ... ; plot the figure
attach_china_map(wks, plot)
draw(plot) ; DO NOT FORGET THIS STEP
```

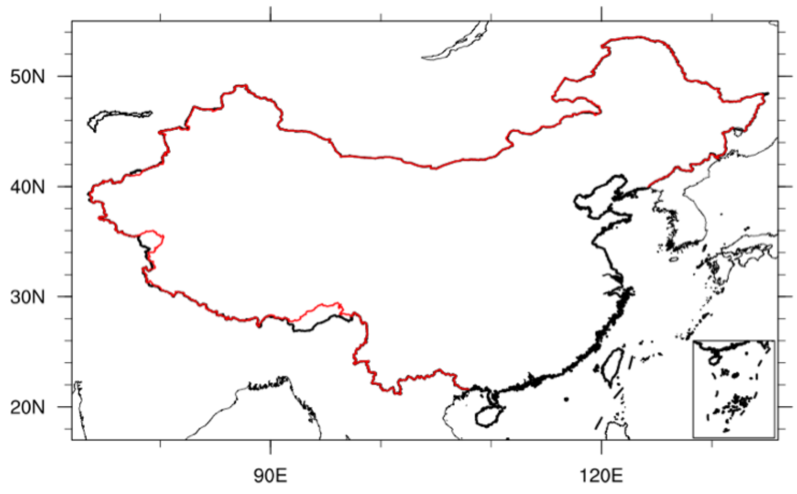

Figure 2.1: The Chinese border in NCL is wrong as shown in red line. The black one is correct.

Due to the high fineness of the border in the shapefile, the plotting speed will be slowed down a bit.

2.2.2 EEMD

2.3 Package usage

The diagnostics platform is composed of several packages. Each package is specialized for one phenomenon or metrics. GEODIAG provides a uniform way to run these packages by

```
geodiag run <package-name> <config-file>
```

`<package-name>` is the name of the package, e.g. `mjo`. `<config-file>` is the configuration file for the package which could be generated by running

```
geodiag config <package-name> [-i]
```

Option `-i` means the command is interactive. Users just need to fill the `<config-file>`. Intermedia data and figures will be generated.

This design makes GEODIAG suitable to post-process model results in batch mode.

3 Developing diagnostics package

GEODIAG is designed as an open platform. A **framework** for writing diagnostics package is specified. When a package is written following the standard of this framework, it can be directly plugged into GEODIAG and readily used by other users. This standard does *not* try to put handcuffs on researchers, but to eliminate the unnecessary duplication of labors.

BASH script is chosen as the developing language, because almost all the servers provide a uniform BASH shell environment. Ruby and Python are more suitable for constructing the framework, but some servers does not provide them[1] and most researchers are not familiar with them. `<package-name>` repesents the package name (e.g. `mjo`) as previous.

The package is driven through `geodiag` command as shown in [Package usage], page 3. Package developers should provide a `driver.sh` BASH script, which contains `<package-name>_help`, `<package-name>_config` and `<package-name>_run` functions so that the package can be treated as a black box. GEODIAG will look for the available packages within the `packages` directory, and check the validity of the packages.

Base function	**Description**
`<package-name>_help`	Print help of the package, especially the options.
`<package-name>_config`	Generate a configuration file.
`<package-name>_run`	Run the package.

Table 3.1: Driver mandatory functions and their meanings.

3.1 Example: MJO package

To present the development of a package, a MJO package is developed as an example.

[1] You will not want to deal with numerous software dependencies to install Ruby or Python with necessary version.