Adding a custom preprocessing step to guru.sh

A short guide to editing the convert() function in your delivered script

---

What this is

guru.sh is generated for you by DataGURU, so its exact contents depend on which variables and files you requested — your copy may contain one or several sections, each built around its own **convert()** function. As delivered, each convert() function is empty (a no-op):

```
function convert {
■:
}
```

If a dataset needs a bit of your own processing before guru.sh regrids it — a unit fix, a masking step, a custom interpolation, whatever your analysis needs — you can edit convert() yourself to call an external script of your own. This guide shows you how.

What to do

1. Make sure the environment variable **my_lpjgmixnmatch** is set to the top directory of your source data (guru.sh already requires this to run at all).

2. In each convert() function you want to hook — there may be just one, or several, depending on what you asked DataGURU for — replace the ":" with:

```
function convert {
■hook="./my_processfilescript.sh"
■if [ -x "$hook" ]; then
■■"$hook" "$@"
■fi
}
```

3. Write your own **my_processfilescript.sh**, make it executable (`chmod +x my_processfilescript.sh`), and place it in the working directory you pass to guru.sh.

If this file is not present, or is not executable, nothing changes — convert() simply does nothing, exactly as it did before you edited it.

What your script receives

convert() calls your script with two positional arguments:

- **$1** — the name of the data file to process, sitting in the current working directory.
- **$2** — the year of that file (guru.sh processes data one year at a time).

Important: variables are not shared

my_processfilescript.sh runs as its own separate process. It does **not** automatically see any variable names used inside guru.sh — only the two arguments above are passed to it. Start your script by naming them yourself, for example:

```
#!/bin/bash
infile="$1"
year="$2"

# ... your processing here, operating on "$infile" ...
```

The result must overwrite the same file

Immediately after calling convert(), guru.sh continues working on the same filename ($1) to regrid it. Your script must therefore write its output back to that same filename (for example, process to a temporary file and then move it over the original) rather than producing a differently named output.

If you edit more than one convert() function

If your guru.sh has several dataset sections and you point more than one of their convert() functions at the same my_processfilescript.sh, that one script will be called once per section, with different $1/$2 each time. Either keep it generic enough to handle every dataset it might see, restrict the edit to just the convert() function(s) you actually want to hook, or have the script check $1 and skip quietly for files it does not recognize.

When you are happy with the result

This is meant for local experimentation only. Once your preprocessing script does what you need, please get in touch with us directly so the logic can be reviewed and incorporated properly — scripts are not picked up or merged automatically.