



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación

R tools users meeting

An-Chi Ho

contributor: Victòria Agudetse

05/05/2022

Agenda

1. Ice-breaker: Which working environment do you use?
2. News
 - startR
 - s2dv
 - R-related issue
3. User presentation: How to source the functions in non-public GitLab repository?
[Victòria]
4. Q&A
 - Can correlation difference use other methods apart from Pearson?
https://earth.bsc.es/gitlab/es/s2dv/-/merge_requests/88#note_165380
 - PlotEquiMap longitude labelling [Roberto]

Ice-breaker

How do you program in R?

Go to minutes google document to answer these questions:

1. Which places do you use to program in R?
2. Which editors do you use to program in R?
3. Why do you choose this kind of environment? What's the advantages and disadvantages you find in these options?

RStudio on WS & Nord4

on WS, open in X11:

- > module load RStudio
- > module load R/3.6.1-foss-2015a-bare #Change from 3.2.0 to 3.6.1
- > rstudio

on Nord4, open in X11:

- > module load RStudio/2021.09.2
- > rstudio

on Nord4, submit in queue and open by URL: (recommended)

Step by step on wiki:

https://earth.bsc.es/wiki/doku.php?id=computing:nord3-v2#using_rstudio-server_in_nord3-v2

VSCoDe on WS

Open VSCoDe module in X11:

```
> module load VSCoDe-stable/1648620864
```

```
> code
```

Install VSCoDe in workstation or laptop:

See Lluís' presentation in the previous meeting

https://earth.bsc.es/wiki/lib/exe/fetch.php?media=tools:r_user_meeting_03032022.pdf

*The two ways are expected to be the same.

*VSCoDe can't be opened on Nord4.

Jupyter Notebook on WS & Nord4

on WS:

- > module load R/3.6.1-foss-2015a-bare
- > module load jupyterlab/1.2.2-foss-2015a-Python-3.7.3
- > jupyter-lab

on Nord3v2, submit in queue and open by URL:

https://earth.bsc.es/wiki/doku.php?id=computing:nord3-v2#using_jupyter_notebooks_in_nord3-v2

startR

New internal release v2.2.0-1

- Installed in workstation_R/3.6.1 and Nord3v2_R/3.6.2. NOT on Nord3v2_R/4.1.2 yet.
- Possible problem: The startR script may be slower because of metadata processing. Please let me know if you detect significant slowdown.
- Pay attention to metadata that are loaded along with data. It may be helpful to the analysis or data check.

Is it good to obtain metadata obligatorily?

- The parameter `return_vars` in Start() is used for obtaining metadata.
- Even with `retrieve = FALSE`, return_vars can load the metadata (i.e., occupy the memory)
- The new development in v2.2.0-1 includes metadata reshaping. It increases the time and memory consumption while running Start().
- It is a good practice obtaining metadata along with data, so we can ensure the correctness and reproducibility of analysis. However, it leads to lower efficiency too.
- Start() mandatorily returns metadata if the selectors are assigned by values or the dimension is reordered/dependent on others.

```
! Warning: All '*_var' params must associate a dimension to one of the requested  
! variables in 'return_vars'. The following variables have been added
```

```
! to 'return_vars': 'time', 'lat', 'lon'
```

```
! Warning: Found 'time' dependency on file dimension 'date', but 'time' is not in  
! return_vars list or does not include 'date'. To provide the correct
```

```
! metadata, 'date' is included under 'time' in 'return_vars.'
```

- Should we keep the option open to users? Downside: Users need to keep an eye on the returned metadata and learn the correct way to define `return_vars`.

s2dv

Clim() accepts dat_dim = NULL

- s2dv::Clim has two data inputs, exp and obs, and it does per-pair climatology. dat_dim could not be NULL before because if so, per-pair method is not applied and it loses the meaning of using this function (simply using mean() can achieve the same purpose).
- However, the function accepts dat_dim = NULL now to be more flexible.
- 'obs' doesn't need to have dat_dim now. So, no need to insert 'member' dimensions for obs data before using Clim().
- The output dimensions could be wrong for some cases. The error has been corrected.

Status: in master branch







Clim() efficiency improvement









s2dv::Clim is improved to have better efficiency. It is still slower and more memory consuming than s2dverification::Clim due to higher flexibility.







Status: in master branch

→ Is there anyone using method 'NDV' and 'kharin'? The efficiency can be further slightly improved if these two methods are removed.

data size: 45.68 Mb

Code	File	Memory (MB)	Time (ms)
new s2dv::Clim	<expr>	-515.4  715.2	2920 
▶ Apply	Clim_clim_isna.R	-515.4  462.9	2360 
InsertDim	Clim_clim_isna.R	0  93.9	170 

Code	File	Memory (MB)	Time (ms)
old s2dv::Clim	<expr>	-600.8  875.3	15950 
▶ MeanDims	Clim_clim.R	-193.6  284.4	13460 
▶ Apply	Clim_clim.R	-407.2  450.0	2200 
InsertDim	Clim_clim.R	0  89.0	240 

Code	File	Memory (MB)	Time (ms)
s2dverification::Clim	<expr>	0  542.6	1350 
Mean1Dim	Clim_s2dverificatio...	0  311.0	760 
na_array <- na_array + as.numeric(is.na(var_exp[i_dat, i_memb, ,	Clim_s2dverificatio...	0  84.2	160 

R-related issue

R-related issue

- IT periodic talk by Albert Vila: it's worth having a look!
https://earth.bsc.es/wiki/lib/exe/fetch.php?media=computing:it_periodic_talks_-_02-05-2022.pdf
- Nord3v2_ **R/4.1.2**-foss-2019b is ready to use!
 - The packages that are in both WS_R/3.6.1 and Nord3v2_R/3.6.2 are installed.
 - Please start migrating your scripts to this module and report any issues you find.
 - WS_R/4.1.2 hasn't had any packages installed yet.
 - We plan to move to R/4.1.2 officially in 6 months.

How to source functions from non-public GitLab repositories?

Sourcing functions in R

In an R script or session, we can import code written in other files that we want to use (e.g. a function) using `source()`.

Sometimes, when we are working with GitLab, it is more practical to **source the files from the online repository**, rather than from our own local copy.

Sourcing functions from a public GitLab repository is very simple! We just need the URL to the 'raw' file:

```
> source('https://earth.bsc.es/gitlab/external/cstools/-/raw/master/R/s2dv_cube.R')
```

However, when we try to source from an **internal** or **private** repository, things get complicated...



Sourcing from a non-public repository

For example, let's try to source the function `get_regrid_params.R` from the CSOperational repository:

```
> source("https://earth.bsc.es/gitlab/external/cstools/-/raw/master/R/s2dv_cube.R")  
# no problem!
```

```
> source("https://earth.bsc.es/gitlab/es/csoperational/-/raw/master/R/  
get_regrid_params.R")  
Error in source("https://earth.bsc.es/gitlab/es/csoperational/-/raw/master/R/  
get_regrid_params.R") :
```

```
https://earth.bsc.es/gitlab/es/csoperational/-/raw/master/R/get_regrid_params.R:1:1:  
unexpected '<'
```

```
1: <  
   ^
```

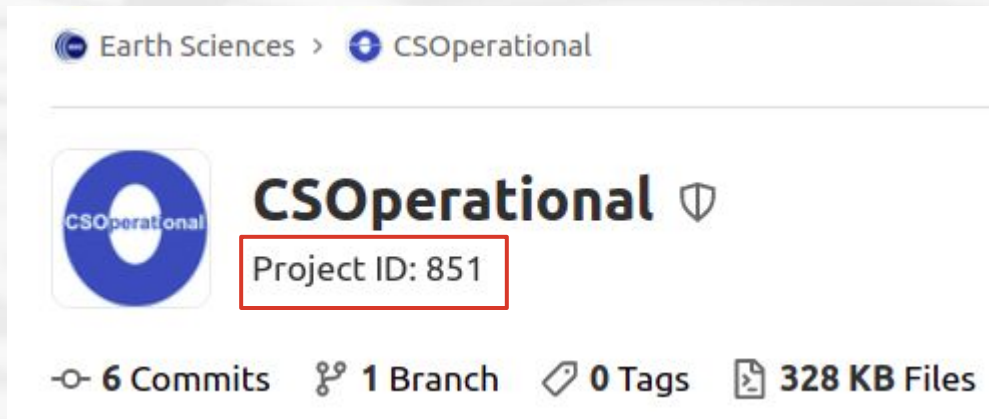
```
# What happened!?
```

The GitLab API: Repository Project ID

GitLab is not returning the raw version of the file! Non-public files can only be downloaded or sourced using the **GitLab API**. The way to access a file through the API is a little different...

First of all, to access any repository through the API, we need its ID number.

We can open our browser and find the repository **Project ID** right underneath its name on GitLab:



The screenshot shows a GitLab repository page for the user 'CSOperational'. At the top, the breadcrumb navigation shows 'Earth Sciences > CSOperational'. Below this is the repository header, which includes the user's profile picture (a blue circle with 'CSOperational' text), the username 'CSOperational' with a shield icon, and a red box highlighting 'Project ID: 851'. At the bottom of the header, there are statistics: '6 Commits', '1 Branch', '0 Tags', and '328 KB Files'.

The GitLab API: File (“blob”) SHA

According to the [GitLab API documentation](#), we can access a certain file through its unique SHA (Simple Hashing Algorithm). An SHA is an alphanumeric string that identifies the file.

Raw blob content

Get the raw file contents for a blob by blob SHA. This endpoint can be accessed without authentication if the repository is publicly accessible.

```
GET /projects/:id/repository/blobs/:sha/raw
```



Supported attributes:

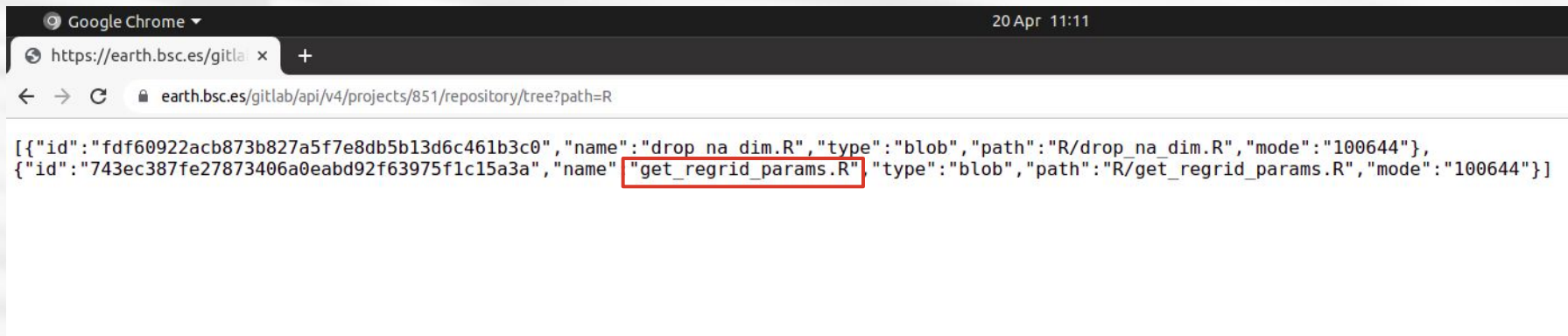
Attribute	Type	Required	Description
<code>id</code>	integer or string	yes	The ID or URL-encoded path of the project owned by the authenticated user.
<code>sha</code>	string	yes	The blob SHA.

How to get your file SHA (I)

We can see the top level files and directories in the repository tree on our browser by typing:

`https://earth.bsc.es/gitlab/api/v4/projects/<project_id>/repository/tree`

If we add “`?path=<path_to_directory>`” at the end of above URL, we will get the contents of our target directory. In this example, the file we want is under a directory named “R”:



```
[{"id": "fdf60922acb873b827a5f7e8db5b13d6c461b3c0", "name": "drop_na_dim.R", "type": "blob", "path": "R/drop_na_dim.R", "mode": "100644"}, {"id": "743ec387fe27873406a0eabd92f63975f1c15a3a", "name": "get_regrid_params.R", "type": "blob", "path": "R/get_regrid_params.R", "mode": "100644"}]
```

How to get your file SHA (II)

Files will have the **"type":"blob"** key-value pair. The value of the "id" key is our file SHA. In this case:

```
"id":"743ec387fe27873406a0eabd92f63975f1c15a3a"
```

Alternative: If you have a local copy of the repository, you can skip this process, open your terminal and simply `cd` to the folder where your file is and type the command `git ls-files -s`:

```
/esarchive/scratch/vagudets/repos/csoperational/R> git ls-files -s
100644 fdf60922acb873b827a5f7e8db5b13d6c461b3c0 0    drop_na_dim.R
100644 743ec387fe27873406a0eabd92f63975f1c15a3a 0    get_regrid_params.R
```

The GitLab API: Personal Access Tokens

The last thing you need to do is generate a [Personal Access Token](#) (PAT). You can do this from your GitLab profile following the steps [in this tutorial](#). Choose either 'api' or 'read_api' (more secure if you only need read access).

Once you have your PAT, you can build your URL by replacing the relevant bits. In summary:

```
https://earth.bsc.es/gitlab/api/v4/projects/<project_id>/repository/blobs/<file_id>/raw?ref=<branch>&private_token=<api_token>
```

[<project_id>](#): Repository "Project ID" number

[<file_id>](#): File SHA

[<branch>](#): The name of the branch you want, e.g. 'master'

[<api_token>](#): Personal Access Token for the GitLab API

Q & A

Can correlation difference use other methods apart from Pearson?

Follow-up for the discussion of `s2dv::DiffCorr` in the last meeting:

- `DiffCorr()`'s correlation difference calculation is based on the methodology in this paper (<https://journals.ametsoc.org/view/journals/mwre/145/2/mwr-d-16-0037.1.xml>), and this method is based on Pearson correlation.
- FYI the function `psych::paired.r` also uses the same method.
- `DiffCorr()` provides three correlation method (Pearson, Kendall, and Spearman) but we don't know if the other two methods are suitable or not.

→ To stay safe, we can include only Pearson method for now.

PlotEquiMap longitude labelling



Thanks for joining

Next meeting: 2nd June 2022 (11 am)