
# Climate Model Assessment Framework

*Scripting environment to share science*

Jérôme Servonnat
CEA Engineer/Researcher at LSCE-IPSL, Saclay/Paris, France
Evaluation of the IPSL coupled model

Agence Nationale de la Recherche ANR - MODÈLES NUMÉRIQUES - CONVERGENCE

# Starting point (2013)

CMIP5 has been a pretty heavy and painful exercise…

We need better tools to face the challenge of CMIP6:
ANR Convergence => IPSL, CNRM, CERFACS

Among our needs, we want to make a significant step in the way we analyse our simulations, and make the evaluation of our models: WP5 Convergence:

**CliMAF core team**: Stéphane Sénési, Jérôme Servonnat, Ludivine Vignon
**+ contributors:** Marie-Pierre Moine, Emilia Sanchez-Gomez, Olivier Marti, Patrick Brockmann, Sébastien Denvil

CliMAF
Sharing – Simplifying – Optimizing

# Gathering users expectations

The specifications to develop CliMAF are the result of:

- two dedicated consultations of the potential users at CNRM/CERFACS and IPSL
- Multiple meetings between the development team
- one year between the first discussion and the first line of code

=> Very important for us to associate the users in defining the specifications of this community tool if we want them to use it

CliMAF
Sharing – Simplifying – Optimizing

# Main requirements

- An easy and common way to access various data trees/organizations (outputs from different models, reference datasets)
- Really simplify those daily pretreatments: selecting a period, a geographical domain, computing a climatology, regrid datasets…
- Do standard plots (maps/curves) and build an html page
- Avoid recomputing! I don't want to compute the same result twice
- Share diagnostics => gathering the local expertise in bigger tools, taking into account that the diagnostics can be written using a variety of languages…

=> Toward a seamless approach: building blocks that will help scientists in their daily work and provide an efficient framework for the more complex/big tools

CliMAF
Sharing – Simplifying – Optimizing

# Technical points in short

- Works on CF-compliant netcdf files
- Produces netcdf files and figures
- Uses CDO for the pre-treatments
- And NCL for the plots
- Smart cache to fully handle outputs and avoid recomputing
- Easy to install (clone of git repo, no compilation):
  https://github.com/senesis/climaf
- Documentation: http://climaf.readthedocs.io/

CliMAF is a 'command line' manager : a scripting facility which allows to launch and pipe user-provided diagnostic scripts (any language) and binaries

CliMAF
Sharing – Simplifying – Optimizing

# Outline

1. Data access
2. Cache management
3. How to add my script
4. The C-ESM-EP

CliMAF
Sharing – Simplifying – Optimizing

# Outline

1. **Data access**
2. Cache management
3. How to add my script
4. The C-ESM-EP

# 1. Data access

In CliMAF we define and access the data with ds() (shortcut to cdataset()):

```
dat1 = ds(   project = 'CMIP5',
             model = 'IPSL-CM5A-LR', experiment = 'historical',
             simulation = 'r1i1p1', variable = 'tas',
             frequency = 'monthly', period = '1980-2005' )
```

**Mandatory** / specific

ds() takes attributes (or facets, or keywords) that define the dataset in a CliMAF 'project':

- Definition of path/filename pattern(s)
- That include a set of keywords (arguments for ds())

CliMAF
Sharing – Simplifying – Optimizing

# 1. Definition of a project

Definition of a project: example with the CMIP5 archive at IPSL:

```
# -- Define the path/filename pattern and include attributes
pattern='/prodigfs/project/CMIP5/output/*/${model}/${experiment}/$
{frequency}/${realm}/${table}/${simulation}/latest/${variable}/${variable}_
${table}_${model}_${experiment}_${simulation}_YYYYMM-YYYYMM.nc'


# -- Declare that the project 'CMIP5' takes the following attributes
cproject('CMIP5', ('frequency','monthly'), 'model', 'realm', 'table',
'experiment', ensemble=['model','simulation'],separator='%')

# -- Finalize the 'CMIP5' project
dataloc(project='CMIP5', organization='generic', url=pattern)
```

# 1. Definition of a project

Definition of a project: example with the CMIP5 archive at IPSL:

```
# -- Define the path/filename pattern and include attributes
pattern='/prodigfs/project/CMIP5/output/*/${model}/${experiment}/$
{frequency}/${realm}/${table}/${simulation}/latest/${variable}/${variable}_
${table}_${model}_${experiment}_${simulation}_YYYYMM-YYYYMM.nc'


# -- Declare that the project 'CMIP5' takes the following attributes
cproject('CMIP5', ('frequency','monthly'), 'model', 'realm', 'table',
'experiment', ensemble=['model','simulation'],separator='%')

# -- Finalize the 'CMIP5' project
dataloc(project='CMIP5', organization='generic', url=pattern)
```

Selecting a period required by the user:
⇒ On the files matching the request, covering the period
⇒ Multiple files: no overlap in time

# 1. Convention/standardization

```
dat1 = ds(   project = 'CMIP5',
             model = 'IPSL-CM5A-LR', experiment = 'historical',
             simulation = 'r1i1p1', variable = 'tas',
             frequency = 'monthly', period = '1980-2005' )
```

**Standard** / specific

Building a community tool involves at some point to propose (smart and community defined) conventions/standards to put the users on the same path:

- minimum set of common attributes (variable, period, simulation, frequency; largely inspired by the CMIP5 data reference syntax) => can be used downstream by the plotting scripts or an automated pre-treatment
- use the CMIP variable names and work with SI Units in CliMAF

```
# -- Variable name alias for 'my_project' (possible offset/scale)
calias('my_project', 'cmip_name', 'var_in_file', offset=273.15 )
```

# 1. Example: access to various projects

```
dat1 = ds(   project = 'CMIP5',
             model = 'IPSL-CM5A-LR', experiment = 'historical',
             simulation = 'r1i1p1', variable = 'tas',
             frequency = 'monthly', period = '1980-2005' )

dat2 = ds(   project = 'IGCM_OUT',
             root = '/ccc/store/cont003/thredds', login = 'p86caub',
             model = 'IPSLCM6', simulation = 'CM605-LR-pdCtrl01',
             frequency = 'seasonal', clim_period = '2020_2029',
             variable = 'tas' )

dat3 = ds(   project = 'ref_climatos',
             variable = 'tas', product = 'ERAInterim' )
```

Access to various data organizations:

- using the same variable names
- And the same units
- Without duplication of data

  ⇒ Can be provided to diagnostics that use the same convention

  ⇒ CliMAF is a soft way for the user to get used to new standards

CliMAF
Sharing – Simplifying – Optimizing

# Outline

1. Data access
2. **Cache management**
3. How to add my script
4. The C-ESM-EP

# 2. Cache management

At this stage, dat1 is only a CliMAF (python) object.

```
In [6]: from climaf.api import *
        dat1 = ds(project = 'CMIP5',
                  model = 'IPSL-CM5A-LR', experiment = 'historical',
                  simulation = 'r1i1p1', variable = 'tas',
                  frequency = 'monthly', period = '1980-2005' )
```

We get the result file (execute the underlying operations) with cfile():

```
In [7]: cfile(dat1)
Out[7]: '/prodigfs/ipslfs/dods/jservon/climaf_cache2/bc756/6d385/06e04/5f531/d3ab0/a2064/330ff/c26c1/b900f/337c5/8ffe5/5.nc'
```

Cache directory (user)

Output path/filename (CliMAF)

CliMAF automatically provides a unique path/filename to the output based the location of the cache directory provided by the user and a hash of the CliMAF expression.

# 2. Cache management

At the same time, CliMAF stores in an index file (in the cache directory) the name of the result with the expression describing the sequence of CliMAF operations that lead to the result :

```
cat /prodigfs/ipslfs/dods/jservon/climaf_cache2/index
```

```
(dp0
S"ccdo(space_average(ds('CMIP5.r1i1p1.tos.2009-2099.-27,-12,205,215.CESM1-BG
C.rcp45.monthly.*.ocean.last')),operator='yearmean')"
p1
S'/prodigfs/ipslfs/dods/jservon/climaf_cache2/5eecb/8a876/22ccd/28870/f0a4f/
62c21/93478/3ab54/2ef0f/3a04b/3d07c/0.nc'
p2
sS"ccdo(space_average(ds('CMIP5.r1i1p1.tos.2009-2099.-27,-12,205,215.CESM1-C
AM5.rcp85.monthly.*.ocean.last')),operator='yearmean')"
p3
S'/prodigfs/ipslfs/dods/jservon/climaf_cache2/8a654/63872/bcb94/e7998/60786/
a4680/19c34/0dd9c/9e760/ee5d4/a124f/6.nc'
p4
```

This way, CliMAF fully documents the provenance of its results

CliMAF
Sharing – Simplifying – Optimizing

# 2. Cache management

```
my_result = ccdo(space_average(ds('CMIP5.r1i1p1.tos.
2009-2099.-27,-12,205,215.CESM1-
BGC.rcp45.monthly.*.ocean.last')),operator='yearmean’)
```

↓

```
cfile(my_result)
```

↓

Scan Index

↓

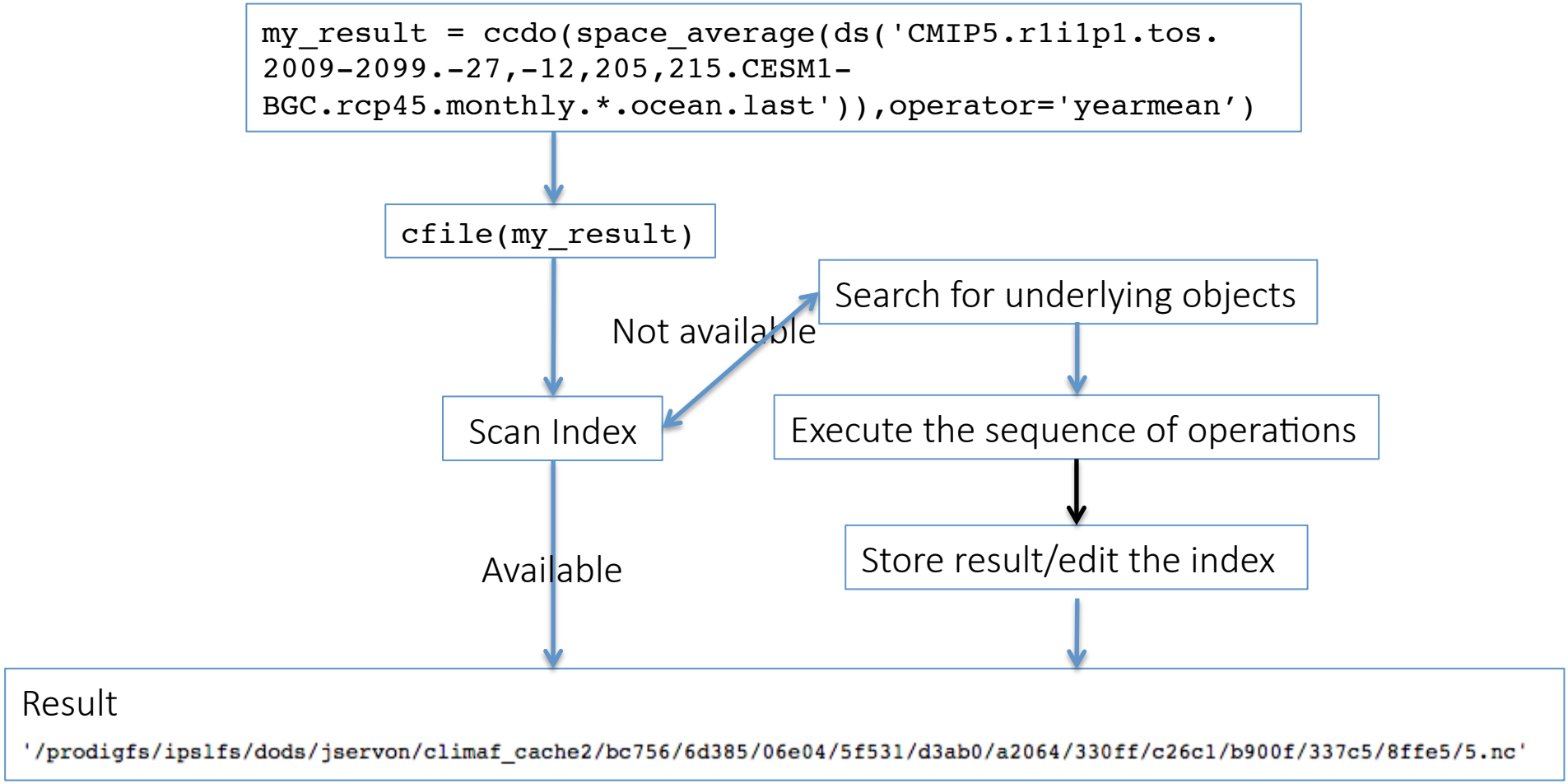
Available

↓

Result

```
'/prodigfs/ipslfs/dods/jservon/climaf_cache2/bc756/6d385/06e04/5f531/d3ab0/a2064/330ff/c26c1/b900f/337c5/8ffe5/5.nc'
```

# 2. Cache management



# 2. Cache management

```
my_result = ccdo(space_average(ds('CMIP5.r1i1p1.tos.
2009-2099.-27,-12,205,215.CESM1-
BGC.rcp45.monthly.*.ocean.last')),operator='yearmean')
```

Save a significant amount of time when executing big tools that:

- crashed for any reason
- or because you want to modify only one plot in the whole set of diagnostics

ing objects

e of operations

Available

Store result/edit the index

Result

```
'/prodigfs/ipslfs/dods/jservon/climaf_cache2/bc756/6d385/06e04/5f531/d3ab0/a2064/330ff/c26c1/b900f/337c5/8ffe5/5.nc'
```

CliMAF
Sharing – Simplifying – Optimizing

# Outline

1. Data access
2. Cache management
3. **How to add my script**
4. The C-ESM-EP

CliMAF
Sharing – Simplifying – Optimizing

# 3. How to add my script?

Any script that:

- runs within a command line
- takes as arguments an input netcdf file and an output netcdf file / figure
- And optional arguments if needed

can become a CliMAF operator => included in the CliMAF framework (data access, cache management…)

```
In [2]: cscript('my_Rscript','Rscript my_script.R ${in} ${out}')
Out[2]: CliMAF operator : my_Rscript
```

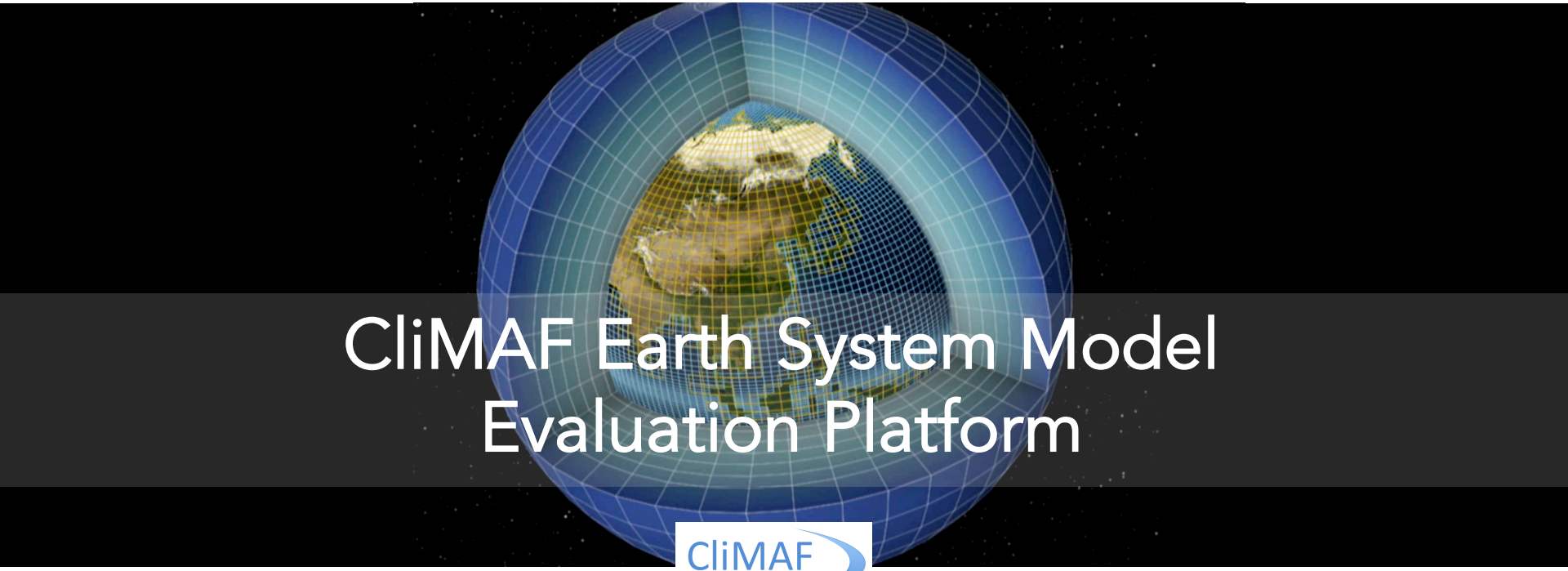
(I have real examples to show you if you are interested)

CliMAF
Sharing – Simplifying – Optimizing

# Outline

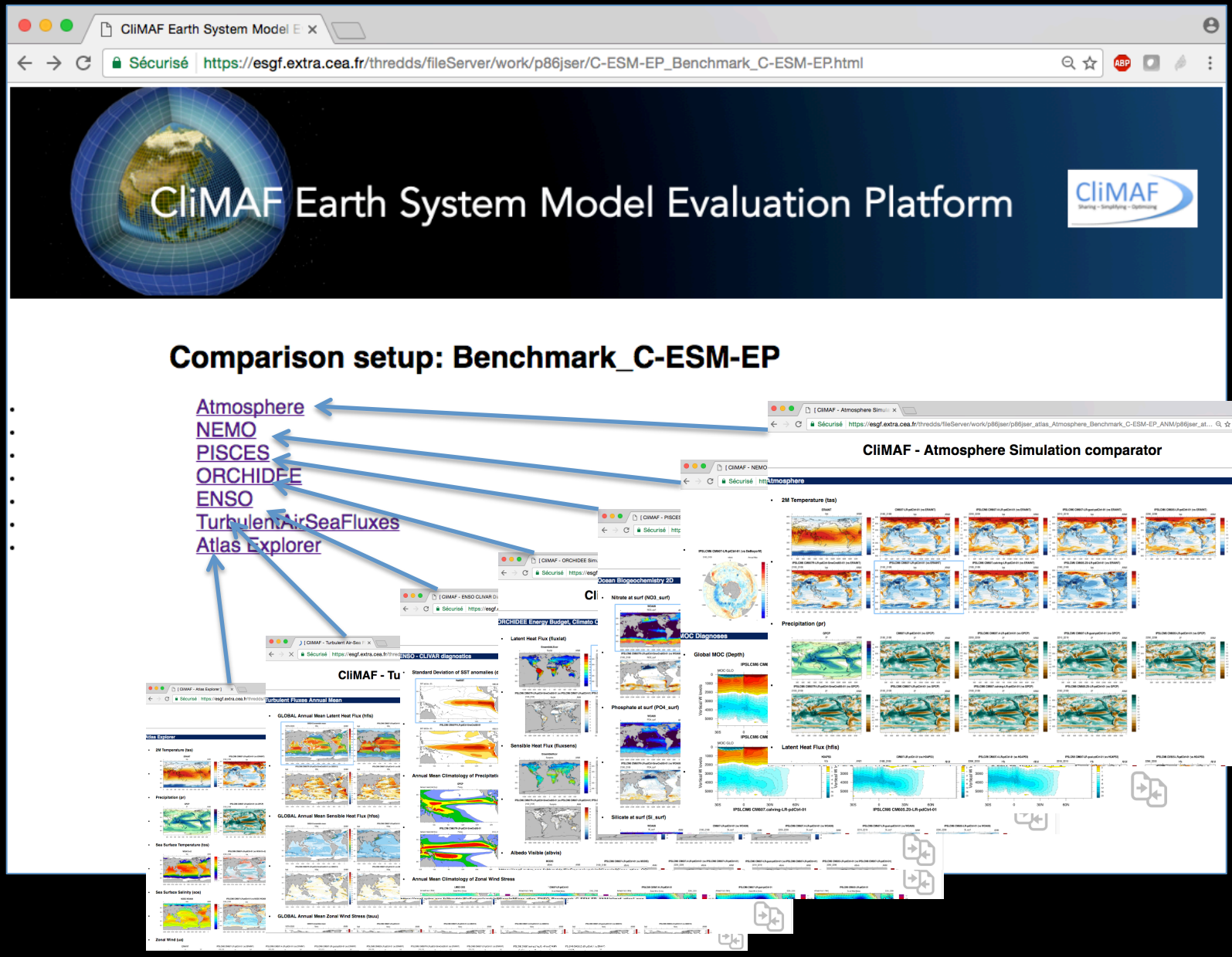
1. Data access
2. Cache management
3. How to add my script
4. The C-ESM-EP

# 4. The C-ESM-EP



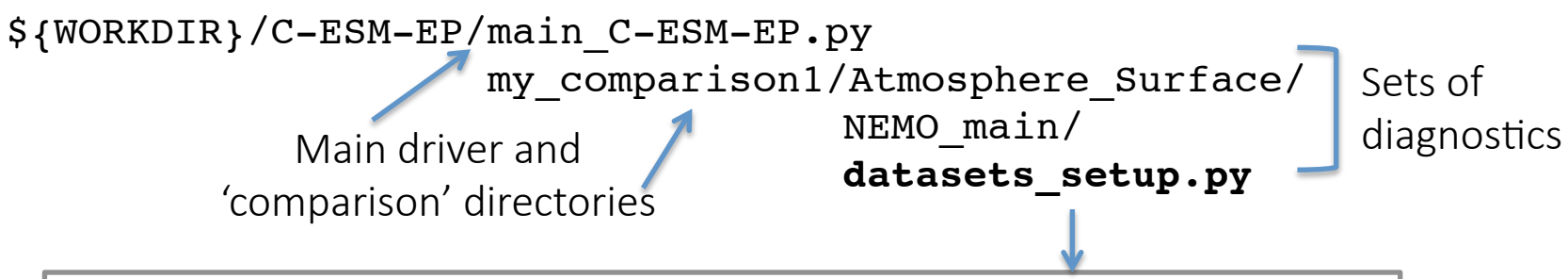
## CliMAF Earth System Model Evaluation Platform

Evaluating/comparing a set of simulations/models at IPSL/CNRM


CliMAF Earth System Model E
Sécurisé https://esgf.extra.cea.fr/thredds/fileServer/work/p86jser/C-ESM-EP_Benchmark_C-ESM-EP.html
CliMAF Earth System Model Evaluation Platform
CliMAF
Comparison setup: Benchmark_C-ESM-EP
Atmosphere
NEMO
PISCES
ORCHIDEE
ENSO
TurbulentAirSeaFluxes
Atlas Explorer
CliMAF - Atmosphere Simulation comparator

# The C-ESM-EP user interface in short

## How to provide my datasets in datasets_setup.py




```
models = [
     dict(project='CMIP5', model='CNRM-CM5',
          experiment='historical', period='1980-2005'),
     dict(project='CMIP5', model='IPSL-CM5A-MR',
          experiment='historical', period='1980-2005'),

     dict(project='IGCM_OUT', model='IPSLCM6',
          simulation='CM6012.1-pd-ttop-02', login='p86caub'),
     dict(project='IGCM_OUT', model='IPSLCM6',
          simulation='CM6012.1-pd-spli-01', login='p86caub'),
]
```

⇒ the user can provide any dataset described by a CliMAF project to the C-ESM-EP

⇒ Python dictionaries = really powerful way to provide instructions to the tool and finess your analysis

CliMAF
Sharing – Simplifying – Optimizing

# Identified limitations of CliMAF

- cache: slows down when overloaded => need to implement a smarter way to clean the cache (we already have some, but still not optimal)

- generates small netcdf files (not so file-system-friendly…)

- specific, refined analyses are not necessarily easy to do => if you can't do what you want with the core CliMAF functionalities, you need to develop your own script and plug it (worth it if you plan to use it routinely)
  => rather use the language you're used to

CliMAF
Sharing – Simplifying – Optimizing

# Where we are now

- a couple of training sessions (CliMAF and C-ESM-EP) at IPSL and CNRM/CERFACS
- we have been really satisfied with CliMAF to build the C-ESM-EP
- more and more scientists start to use it to build their own data processing routines
- Kind of old-school but very easy to understand for a scientist who is not really into object-oriented programming

CliMAF
Sharing – Simplifying – Optimizing

# Where we are now

- More than the code itself (not so much manpower at the moment…), we are interested to share our experience and ideas => services for the users

CliMAF
Sharing – Simplifying – Optimizing

# Where we are now

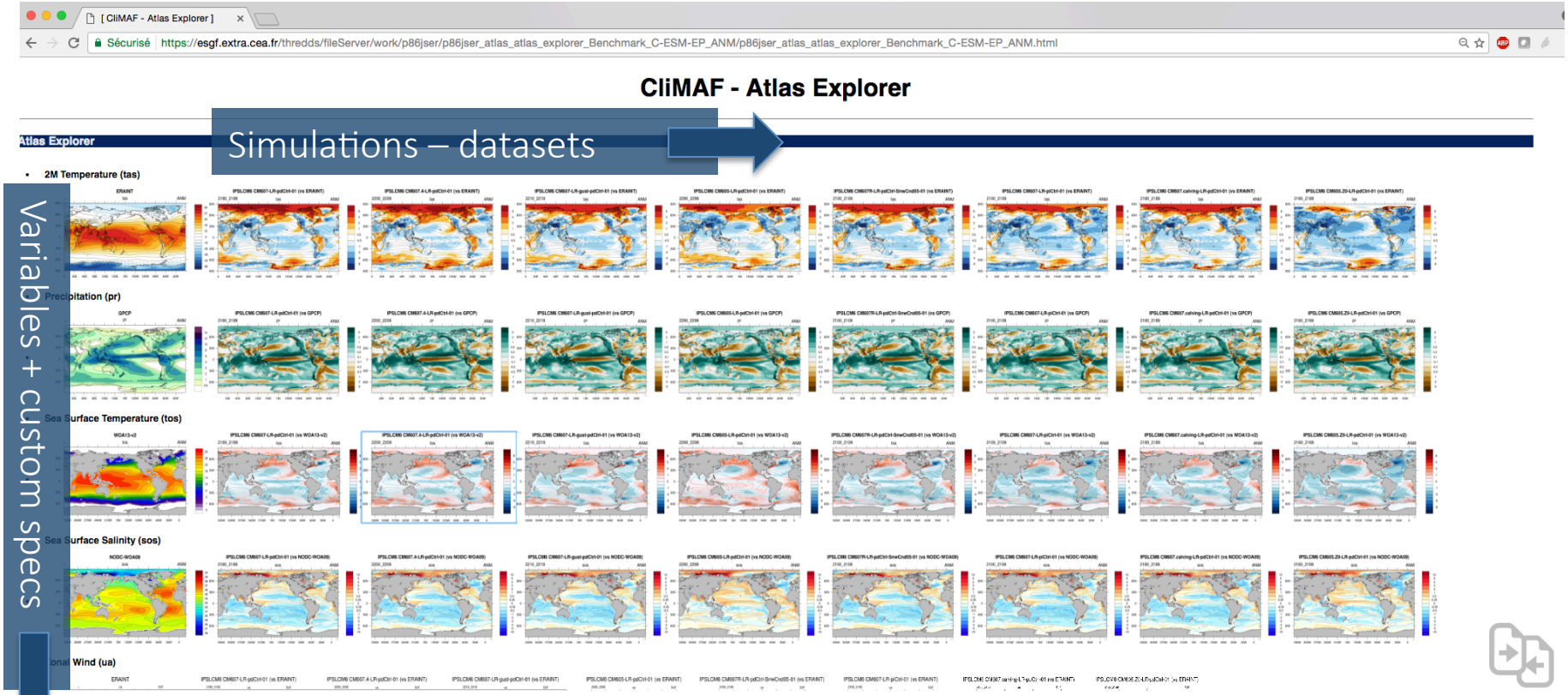
- More than the code itself, we are interested to share our experience and ideas in terms of services for the users

Thank you for your attention!
Questions?

jerome.servonnat@lsce.ipsl.fr

# Atlas Explorer

## Climatology / difference maps on demand

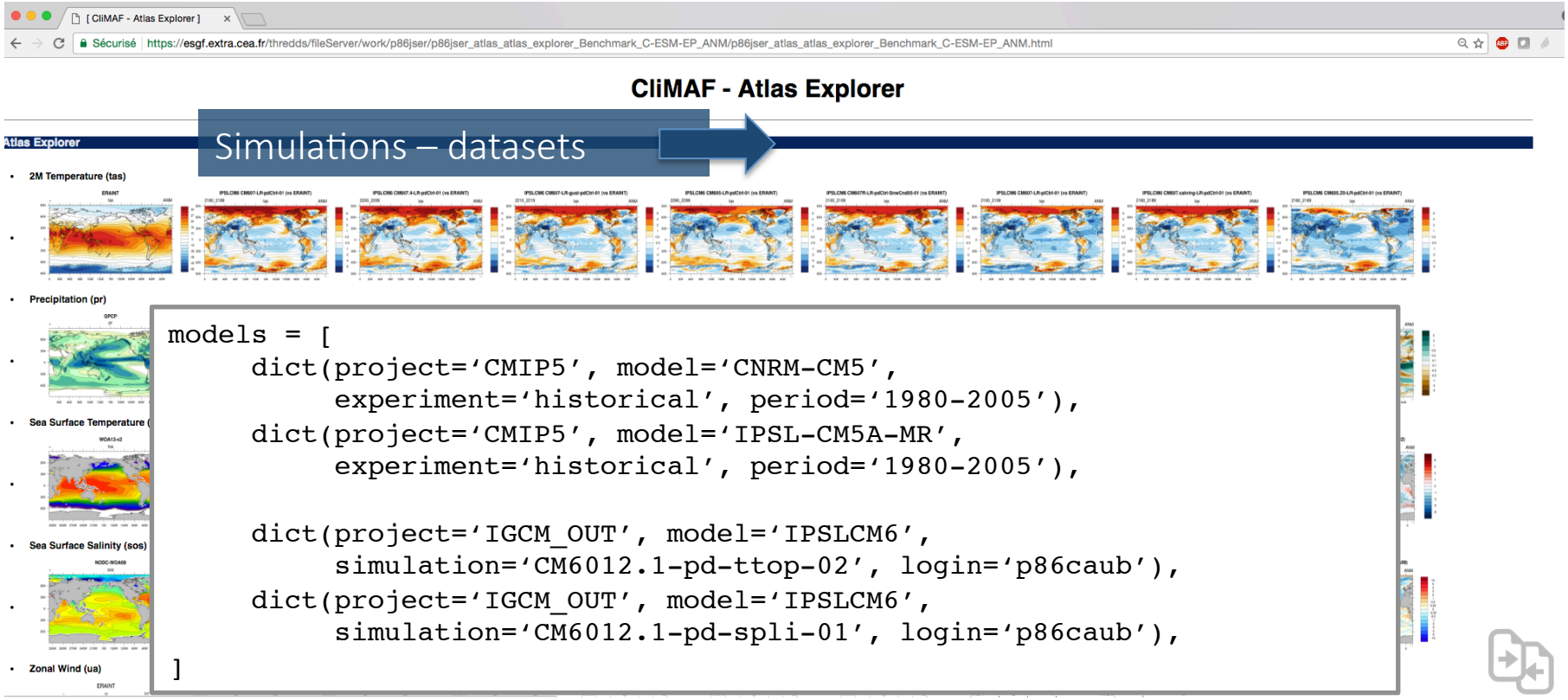



Two python parameter files, containing lists of dictionaries:

- One for the datasets (datasets_setup.py)
- One for the variables + customs specs

# The C-ESM-EP user interface

## How to provide my datasets in datasets_setup.py




⇒ the user can provide any dataset described by a CliMAF project to the C-ESM-EP

CliMAF
Sharing – Simplifying – Optimizing

# 1. Using python dictionaries

Interesting feature: we can use a python dictionary to provide the attributes / keywords to ds():

```
dat_dict = dict( project = 'CMIP5',
                 model = 'IPSL-CM5A-LR', experiment = 'historical',
                 simulation = 'r1i1p1', variable = 'tas',
                 frequency = 'monthly', period = '1980-2005' )
dat = ds(**dat_dict)
```

Consequence: If you make a python list of dictionaries to specify multiple datasets, you are highly flexible in the content of the dictionaries (update, pop)