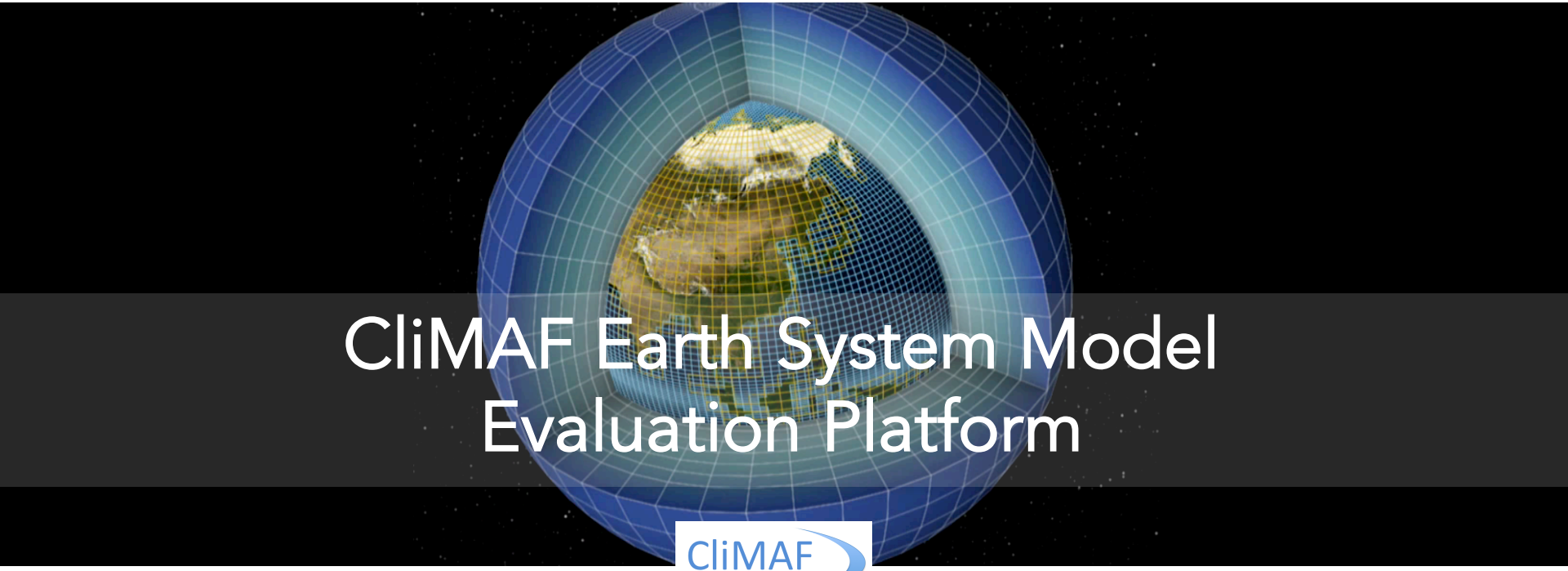


CliMAF Earth System Model Evaluation Platform

CliMAF
Sharing – Simplifying – Optimizing

Evaluating/comparing a set of simulations/models on Ciclad:
Demo / TP

Agence Nationale de la Recherche ANR - CONVERGENCE MODÈLES NUMÉRIQUES

Outline

C-ESM-EP Demo / Practicals

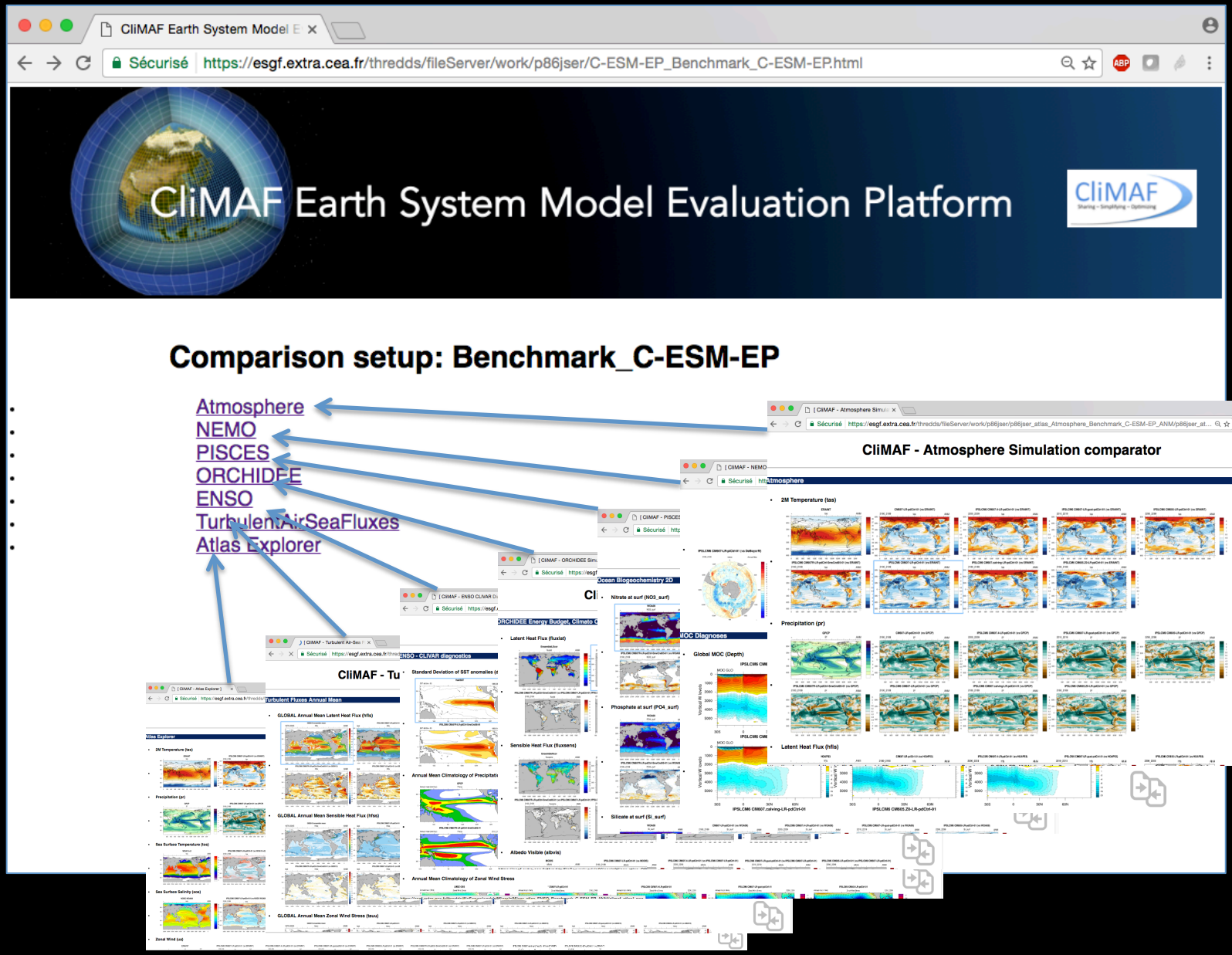
1. **Running the C-ESM-EP demo for all components**

2. **Atlas Explorer as an exploratory tool:**

    1. **Modify the input datasets:**
        1. Add/modify datasets (CMIP5 models or IGCM_OUT simulations)
        2. Change the period and frequency
        3. Setup your own CliMAF project
        4. Control the reference

    2. **Using the Atlas Explorer parameter file**
        1. Overall parameters
        2. Play with the dictionaries of the variables
        3. Add new variables
        4. Modify the plot parameters

1. Running The C-ESM-EP


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
2M Temperature (tas)
Precipitation (pr)
Latent Heat Flux (hfls)

The C-ESM-EP in a nutshell

Prerequesite on your data

1. The data you want to work on is described by a CliMAF 'project' (data tree + variable names aliases)

2. The data has to be CF-compliant (norm on the metadata in the files, especially the dimensions)

3. We encourage working with CMIP variable names

4. No overlapping files in your data directories (do not have multiple files covering a common period for the same variable in the same data tree)

Getting started
Web server

Note: running the C-ESM-EP implies storing the results on a space that can be reached from the web.

Ciclad:
/prodigfs/ipslfs/dods/${username}

Curie:
/ccc/work/cont003/thredds/${username} #(if working on gencmip6)
/ccc/work/cont003/dods/public/${username} #(if working on dsm)

If you don't have an account on those spaces, send a request to:

- Ciclad: svp-ciclad@ipsl.jussieu.fr
- TGCC: hotline.tgcc@cea.fr

Getting started

Getting the C-ESM-EP sources folder

- Ciclad:

  sources=/home/jservon/C-ESM-EP/src

- Curie:

  sources=/ccc/cont003/home/dsm/p86jser/C-ESM-EP/src

For both Ciclad and Curie, we advise to work on /home to take advantage of the backup.

The C-ESM-EP in a nutshell

The quick way to use the C-ESM-EP on Ciclad

1. Copy the sources in a working directory:

```
cd my_working_directory
mkdir -p C-ESM-EP ; cd C-ESM-EP
cp –r ${sources} . ; cp –r src work
cd work
```

2. Setup your comparison:

```
cp -r comparison_example/ all_components_demo/
```

3. Run all the components together or just a subset:

```
python run_C-ESM-EP.py all_components_demo
```

5. See the results on the URL returned by run_C-ESM-EP.py

```
-- The CliMAF ESM Evaluation Platform will be available here:
--
--   https://vesg.ipsl.upmc.fr/thredds/fileServer/IPSLFS/jservon/C-ESM-EP/all_components_demo_jservon/C-ESM-EP_all_components_demo.html
--
--
-- html file can be seen here:
-- /prodigfs/ipslfs/dods/jservon/C-ESM-EP/all_components_demo_jservon/C-ESM-EP_all_components_demo.html
(PMP_nightly-nox) jservon@ciclad-ng:~/C-ESM-EP/work>
```

6. If you only want the URL, run:

```
python run_C-ESM-EP.py all_components_demo url
```

Getting started

Output directory for the results / portable atlas

On Ciclad, we can write directly on the dods server. Thus, the output directory is automatically set to:
CESMEP_OUT=/prodigfs/ipslfs/dods/${USER}

The result of the C-ESM-EP is meant to be portable, i.e. you can copy the top directory of your comparison on any space that you can access with a web browser and you will be able to read it.

The output tree is:
${CESMEP_OUT}/C-ESM-EP/${comparison}_user/

- C-ESM-EP_${comparison}.html
- **${component}_${season}/**
  - atlas_${component}_${comparison}_${season}.html
  - climaf_atlas*.png

In every ${component}_${season} directories you will find the html file of the atlas of the component, and the png images coming with it.

2. Playing with Atlas Explorer

Atlas Explorer
Goal - philosophy

Atlas Explorer is an easy and flexible way to produce an html page showing climatologies and difference maps (with a reference) on a set of datasets (simulations, models, different periods…).

The main goal of the Atlas Explorer is to provide the user with a set of predefined features to assess a variable (plotting parameters, default observational reference) while keeping a lot of control on the diagnostics from only one parameter file.

In this demo/practical, we will see how to control:
- the input datasets (CMIP5 models, your simulations…)
- The diagnostics (variables, region, season, customising the plots…)

Getting started with Atlas Explorer

Setup for an interactive session

Submit an interactive session with:

```
qsub –IV –q std
cd my_working_directory/C-ESM-EP/work
```

Setup a comparison directory to play with Atlas Explorer:

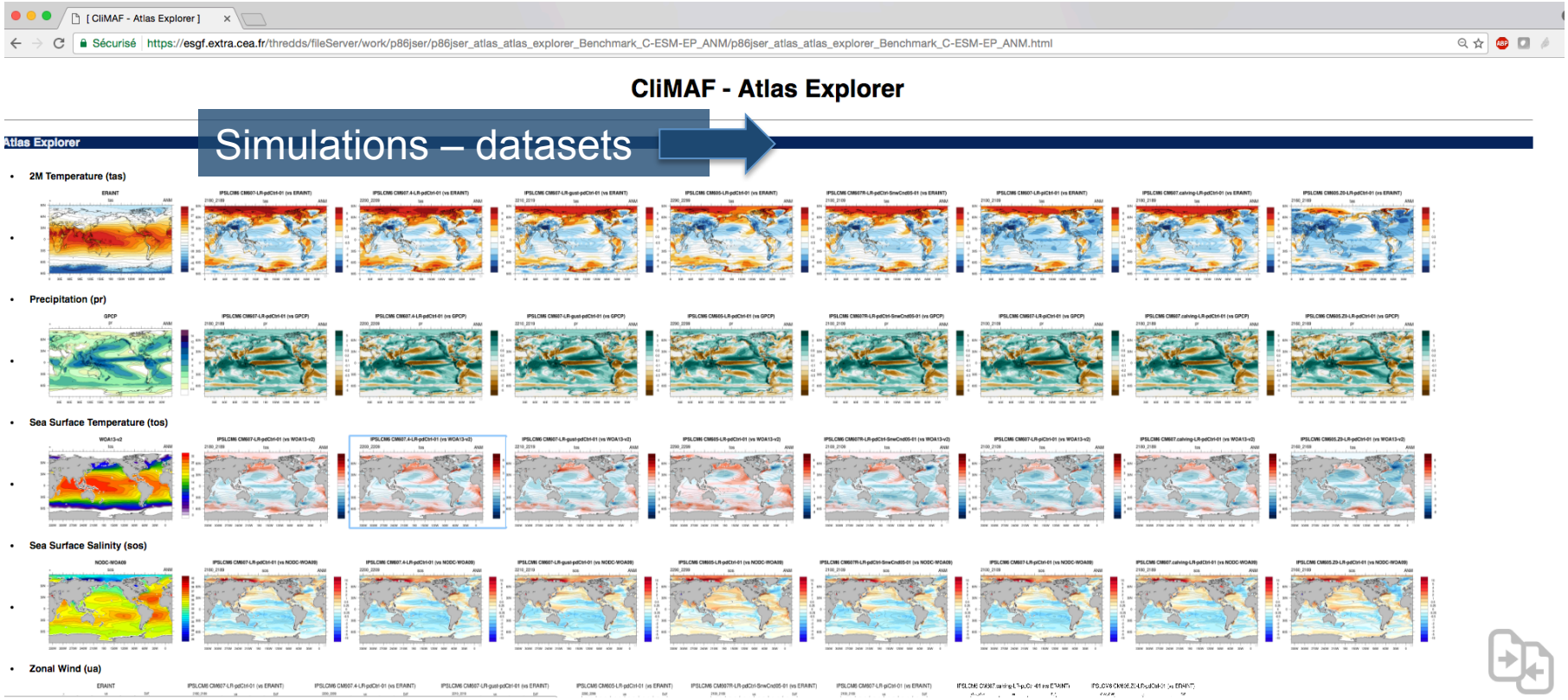
```
cp -r comparison_example/ toy_comparison/
cd toy_comparison
```

You're all set to start!

2.1 Add/modify datasets

2.1 Add/Modify the datasets

datasets_setup.py



2.1 Add/Modify the datasets

datasets_setup.py

The datasets are specified in **datasets_setup.py**

The datasets are defined as python dictionaries containing the keywords/values pairs that are typically used by the CliMAF function ds() to access the data (but the variable).

The dictionaries can contain some custom control on the diagnostics (provide a season, a period and frequency specific to a diagnostic…).

Let's start by opening datasets_setup.py with your favourite editor:

```
vi datasets_setup.py
```

We will now learn how to add or modify the datasets in datasets_setup.py, using the pre-existing **CliMAF projects (data access definitions).**

2.1.1 CliMAF datasets

Standardized interface to access datasets

CliMAF provides a **standardized way to access different data structures**, defined with **CliMAF projects**, with a set of **common/mandatory keys** and possible **specific keys** to describe the data structure.

We access the datasets with the function ds() (shortcut to cdataset()) by providing **pairs of keywords/values**. Example (in CliMAF code):

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

Mandatory / specific

This functionality allows **defining variable name aliases** (potential scale/ offset), so that you can work **on all the different data structures with CMIP variables names and SI units.**

2.1.1 The CliMAF projects

Getting access to the various data structures

In CliMAF, we get access to the datasets through **CliMAF 'projects'**.

A CliMAF project is defined by:

- a **pattern** to access the data
- a set of **keywords that define the dataset** (path + filename + variable)
- a set of aliases (variables and frequency) between the **names in the files** and the **CMIP standard names**

Have a look at the documentation here:
http://climaf.readthedocs.io/en/latest/functions_data.html?highlight=cproject

And/or look at the examples (projects already defined in CliMAF) in the CliMAF version used for the C-ESM-EP:

```
/home/jservon/Evaluation/CliMAF/climaf_installs/climaf_1.0.3_CESMEP/
climaf/projects
```

The igcm_out.py project file is a good example.

2.1.1 The 'CMIP5' CliMAF project

Example with the 'CMIP5' data structure

If you set this in datasets_setup.py:

```
models = [
   dict(
            project = 'CMIP5',
            model = 'IPSL-CM5A-LR',
            experiment = 'historical',
            simulation = 'r1i1p1',
            frequency = 'monthly',
            period = '1980-2005'
       )
]
```

You will make use of this pattern:

```
/prodigfs/project/CMIP5/output/*/${model}/${experiment}/${frequency}/
${realm}/${table}/${simulation}/${version}/${variable}/
${variable}_${table}_${model}_${experiment}_${simulation}_${period}.nc
```

To access this file

```
/prodigfs/project/CMIP5/output/IPSL/IPSL-CM5A-LR/historical/mon/
atmos/Amon/r1i1p1/latest/tas/
tas_Amon_IPSL-CM5A-LR_historical_r1i1p1_185001-200512.nc
```

2.1.1 The 'CMIP5' CliMAF project

Example with the 'CMIP5' data structure

The available keys and default values are:

```
model      =  no default
experiment = 'historical'
realm      = '*'
table      = '*'
simulation = 'r1i1p1'
version    = 'last'
frequency  = 'monthly'
period     =  no default
```

You need to define model and period.

Frequency='seasonal' is not available on project CMIP5.

2.1.1 The 'IGCM_OUT' project

Example with the 'IGCM_OUT' data structure

The 'IGCM_OUT' CliMAF project provides access to the data structure of the libIGCM model outputs, like LMDz, ORCHIDEE, NEMO, PISCES (and subsequently, LMDZ-OR, and IPSLCM models)

Here is a typical example of an IGCM_OUT dataset definition:

```
models = [
   dict(project = 'IGCM_OUT',
        root = '/ccc/store/cont003/thredds',
        login = 'p86caub',
        model = 'IPSLCM6',
        simulation = 'CM605-LR-pdCtrl01',
        frequency = 'seasonal',
        clim_period = '2020_2029'
       )
]
```

2.1.1 The 'IGCM_OUT' project

Example with the 'IGCM_OUT' data structure

CliMAF is searching your file among those patterns:

```
${root}/${login}[/IGCM_OUT]/${model}/${status}/${experiment}/
${simulation}/${DIR}/${OUT}/${ave_length}/
${simulation}_YYYYMMDD_YYYYMMDD_${frequency}_${variable}.nc

${root}/${login}[/IGCM_OUT]/${model}/${status}/${experiment}/$
{simulation}/${DIR}/${OUT}/${frequency}${clim_period_length}/
${simulation}_${frequency}_${clim_period}_1M_${variable}.nc
```

1.2.2/ project IGCM_OUT

Available keys

The available keys and default values are:

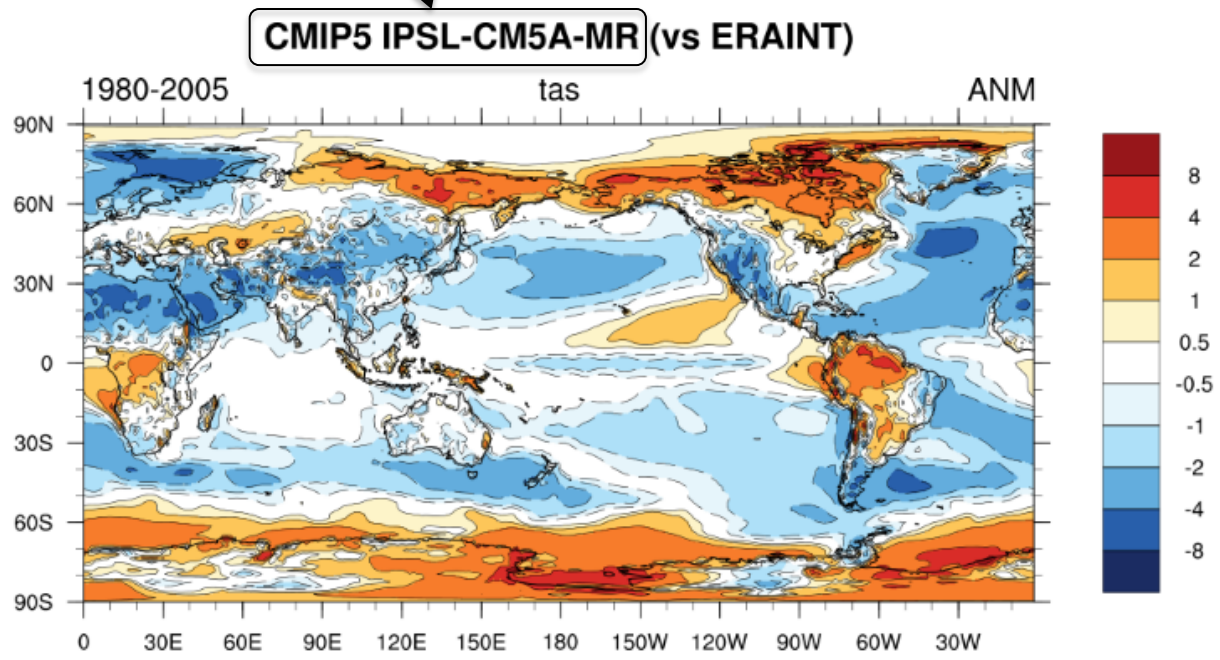
```
- root = '/ccc/store/cont003/thredds' # path (without the login)
                                      # to the top of the IGCM_OUT tree
- login = '*' # login of the producer of the simulation
- model = '*' # explicit
- experiment = '*' # piControl, historical, amip...
- status = '*' # DEVT, PROD, TEST
- simulation = '*' # name of the numerical simulation
                   # (JobName in the IGCM syntax)
- DIR = '*' # ATM, OCE, SRF...
- OUT = 'Analyse' # or 'Output'
- frequency = 'monthly' # daily, annual_cycle (equivalent to 'seasonal')
- ave_length = 'MO' # MO, DA (optionnal, but can reduce the
                    # duration of the localization by ds() )
- period = 'fx' # explicit
- clim_period = '????_????' # explicit
- clim_period_length = '' # can be set to '_50Y' or '_100Y' to access
                          # the annual cycles averaged over 50yr long
                          # or 100yr long periods
```

2.1.1 Name in the plot

Bonus: provide a customname to be used in the plots

Customname: you can provide a customname in the dataset dictionary that will be used in the plots

```
dict(project = 'CMIP5', model='IPSL-CM5A-MR', experiment='historical',
     frequency='monthly', period='1980-2005',
     customname='CMIP5 IPSL-CM5A-MR'
     ),
```



2.1.1 Exercise

Add / modify the datasets

Using the example, add another CMIP5 model and another IGCM_OUT simulation.

Then, run Atlas Explorer:

```
./job_C-ESM-EP.sh AtlasExplorer
```

See the results on the address at the end of the execution.

2.1.1 Debugging

Tracking down the errors

The C-ESM-EP has a 'safe_mode' that returns an empty (white) image when it doesn't successfully produce the plot (by default).

You can track down the origin of the error by setting **'safe_mode'=False** in the associated parameter file in:

```
vi toy_comparison/AtlasExplorer/params_AtlasExplorer.py
```

Also ensure that you set **verbose='debug'** so that CliMAF will write what it's doing (and the details of the error).

2.1.1 Debugging

Tracking down the errors

Example (from a params_AtlasExplorer.py file:

```
# -- Preliminary settings: import module, set the verbosity and the 'safe mode'
# ------------------------------------------------------------------------ >
from os import getcwd
# -- Set the verbosity of CliMAF (minimum is 'critical', maximum is 'debug', intermediate -> 'warning')
verbose='debug'
# -- Safe Mode (set to False and verbose='debug' if you want to debug)
safe_mode = False
# -- Set to 'True' (string) to clean the CliMAF cache
clean_cache = 'False'
```

2.1.2 Control the period

Period selection in brief in CMIP5 and IGCM_OUT

The C-ESM-EP essentially works **on 'monthly' and 'seasonal' datasets.**

With the **'monthly' datasets**, CliMAF can perform a **smart period selection** on the available netcdf files.

With the **'seasonal' datasets**, CliMAF only selects an **existing seasonal cycle** netcdf file. The C-ESM-EP, however, has a smart way to **select the last/first seasonal cycle available**.

Both CMIP5 and IGCM_OUT projects have monthly datasets.
Only IGCM_OUT (and the ref_climatos project) **provides seasonal datasets.**

2.1.2 Control the period

Period selection in the CMIP5 project

One of the great features of CliMAF is the automatic selection of the period provided by the user.

When you set **frequency='monthly'**, you have to provide a period that will be extracted among the non-overlapping netcdf files in the targeted directory.

For instance, this dataset definition:

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

Will extract the period 1980-2005 from this file found in the archive:

```
/prodigfs/project/CMIP5/output/IPSL/IPSL-CM5A-LR/historical/
mon/atmos/Amon/r1i1p1/latest/tas/
tas_Amon_IPSL-CM5A-LR_historical_r1i1p1_185001-200512.nc
```

2.1.2 Control the period and frequency

Period selection in IGCM_OUT

In the IGCM_OUT project, you will have the same behaviour as for the CMIP5 project with **frequency='monthly'** (fundamental CliMAF functionality).

By default, it will target the 'TS_MO' time series files created by the post-processing (one variable per file with all time steps).

If you want to get the data from the 'Output/MO' files, add OUT='Output' to the list of arguments :

```
dict(project = 'IGCM_OUT', root = '/ccc/store/cont003/thredds',
     login = 'p86caub', model = 'IPSLCM6',
     simulation = 'CM605-LR-pdCtrl01'
     frequency = 'monthly', OUT='Output', period = '2020_2029')
```

2.1.2 Control the period and frequency

Working with pre-computed annual cycles IGCM_OUT 'SE'

The IGCM_OUT project allows access to the pre-computed **'/Analyse/SE'** annual cycles provided as a post-treatment.

We access the SE with **frequency='seasonal'**.

Because we don't do period selection on annual cycles, the **'period**' argument is automatically set to **'fx'** (for 'fixed field').

Instead, we use **'clim_period'** that can be:

- an explicit character string providing the pre-existing period
- 'last_SE' or 'first_SE' to get the last/first available annual cycle of the simulation

```
dict(project = 'IGCM_OUT', root = '/ccc/store/cont003/thredds',
     login = 'p86caub', model = 'IPSLCM6',
     simulation = 'CM605-LR-pdCtrl01'
     frequency = 'seasonal', clim_period = '2020_2029'),
dict(project = 'IGCM_OUT', root = '/ccc/store/cont003/thredds',
     login = 'p86caub', model = 'IPSLCM6',
     simulation = 'CM605-LR-pdCtrl01'
     frequency = 'seasonal', clim_period = 'last_SE')
```

2.1.2 clim_period and ts_period

Period for the clim. and the time series C-ESM-EP diagnostics

The C-ESM-EP is composed of **'climatology diagnostics'** (like bias maps) and **'time series diagnostics'** (like the AMOC index or ENSO diagnostics).

Because of this, the user might want to provide **one period for the climatologies, and one for the time series**.

The diagnostics of the Atlas Explorer are only 'climatology diagnostics'. We will describe how to handle the 'time series diagnostics' also here, but it will not be applicable to the Atlas Explorer.

2.1.2 clim_period and ts_period

Period for the clim. and the time series C-ESM-EP diagnostics

Trivial case

If the user sets **frequency='monthly'** and provides a **'period'**, the C-ESM-EP will use this period for both the time series diagnostics and the climatologies.

Separate control of the clim and ts period

The C-ESM-EP has a specific functionality that adds to the CliMAF data access and allows controlling the period for the different diagnostics:

- **clim_period** for the **climatological diagnostics**
- **ts_period for the time series diagnostics**

2.1.2 ts_period

Period for the time series diagnostics

If the user provides a **ts_period** the C-ESM-EP will **automatically set frequency='monthly'** for the diagnostics labeled 'TS'.

ts_period takes the following values:

- an explicit period (ex: '1980-2005', '198001_200012', '20000101-20051231')
- last_XXY / first_XXY to extract the last/first XX years available
- 'full' to get the whole available period

```
dict(project='IGCM_OUT',
     login='p86maf',
     model='IPSLCM6',
     experiment='pdControl',
     simulation='CM608P-LR-pdCtrl-ICEPLUS-01',
     clim_period='1990_1999',
     ts_period='full',
     customname='CM608P.ICEPLUS-01 1990_1999'
     ),
```

2.1.2 clim_period

Period for the climatology (notably for Atlas Explorer)

If the user provides a **clim_period, we have two cases:**

Case 1: frequency='seasonal'

In this case (only for IGCM_OUT), CliMAF will get the 'SE' files.
Possible values (as said previously):

- an explicit character string providing the pre-existing period
- 'last_SE' or 'first_SE' to get the last/first available annual cycle of the simulation

Case 2: frequency='monthly'

If the user provides a 'clim_period' with frequency='monthly', the C-ESM-EP will use the value of clim_period as a 'period'. CliMAF will then handle the period extraction and the C-ESM-EP will compute the climatology over this period.
Possible values:

- an explicit period (ex: '1980-2005', '198001_200012', '20000101-20051231')
- last_XXY / first_XXY to extract the last/first XX years available
- 'full' to get the whole available period

2.1.2 Exercise

Change the period in Atlas Explorer

Using the example, add other periods of the same CMIP5 model and IGCM_OUT simulation to the list of datasets.

Then, run Atlas Explorer:

```
./job_C-ESM-EP.sh AtlasExplorer
```

See the results on the address at the end of the execution.

```
 --
 --
 --
index actually written as : /prodigfs/ipslfs/dods/jservon/C-ESM-EP/toy_comparison_jservon/Atl
asExplorer_ANM/atlas_AtlasExplorer_toy_comparison_ANM.html
may be seen at https://vesg.ipsl.upmc.fr/thredds/fileServer/IPSLFS/jservon/C-ESM-EP/toy_compa
rison_jservon/AtlasExplorer_ANM/atlas_AtlasExplorer_toy_comparison_ANM.html
(PMP_nightly-nox) jservon@ciclad-ng:~/C-ESM-EP/src/toy_comparison>
```

2.1.3 Add your own project

Access your own datasets

You can easily access your own datasets by creating a CliMAF project directly in datasets_setup.py.

You will find an example **in datasets_setup.py with a 'CMIP5_bis' project (same as next slide)**.

The minimum you have to do is a 'cproject' and a 'dataloc'.
You will also see how to use cdef() (define default values) and cfreqs() (aliases for the frequency names).

You're also invited to look at the CliMAF documentation for the use of those functions, and get some inspiration from the CliMAF projects in:

```
/home/jservon/Evaluation/CliMAF/climaf_installs/
climaf_1.0.3_CESMEP/climaf/projects
```

2.1.3 Add your own project

Add access to your own data structure

```
# -- Declare a 'CMIP5_bis' CliMAF project (a replicate of the CMIP5 project)
# ------------------------------------------------------------------------ >
cproject('CMIP5_bis', ('frequency','monthly'), 'model', 'realm', 'table', 'experiment', ensemble=['model','simulation'],
separator='%')
# --> systematic arguments = simulation, frequency, variable
# -- Set the aliases for the frequency
cfreqs('CMIP5_bis', {'monthly':'mon'})
# -- Set default values
cdef('simulation'   , 'r1i1p1'       , project='CMIP5_bis')
cdef('experiment'   , 'historical'   , project='CMIP5_bis')
cdef('table'        , '*'            , project='CMIP5_bis')
cdef('realm'        , '*'            , project='CMIP5_bis')
# -- Define the pattern
pattern="/prodigfs/project/CMIP5/output/*/${model}/${experiment}/${frequency}/${realm}/${table}/${simulation}/latest/${v
ariable}/${variable}_${table}_${model}_${experiment}_${simulation}_YYYYMM-YYYYMM.nc"
# --> Note that the YYYYMM-YYYYMM string means that the period is described in the filename and that CliMAF can
# --> perform period selection among the files it found in the directory (can be YYYY, YYYYMM, YYYYMMDD).
# --> You can use an argument like ${years} instead if you just want to do string matching (no smart period selection)

# -- call the dataloc CliMAF function
dataloc(project='CMIP5_bis', organization='generic', url=pattern)
# ------------------------------------------------------------------------ >
```

2.1.4 Choose the reference

Reference used for the difference maps

The goal of Atlas Explorer is to do **difference maps between a set of datasets** (specified in the 'models' list) **and a reference**, controlled with 'reference' in datasets_setup.py (also in the parameter files for custom control).

The **'default' reference** provides access to a set of pre-defined references (observations, reanalyses) for each variable. This is done via the CliMAF function variable2reference() (not seen by the user) that returns a dataset dictionary defining the reference dataset for the chosen variable.

```
# -- Set the reference against which we plot the diagnostics
# ------------------------------------------------------------------------- >
# --    -> 'default' uses variable2reference to point to a default
# --       reference dataset (obs and reanalyses)
# --    -> you can set reference to a dictionary that will point any other
# --       climaf dataset
# --       For instance, you can set it to models[0] if you want to see the
# --       differences relative to the first simulation of the list 'models'
reference = 'default'
```

2.1.4 Add your reference

Adding user reference to the default references

It is possible to add your own references (if described by a CliMAF project) with custom_obs_dict (either from datasets_setup.py or the parameter file).
It is particularly useful if you defined new variables, or want to change the reference that is provided by default.

Example: specifying the ERAINT reference of the project ref_climatos for a list of variables:

```
# -- Add your custom observations (or a reference for a nbew derived variable)
# ------------------------------------------------------------------------- >
custom_obs_dict = {
   'ua_Atl_sect': dict(project='ref_climatos', product='ERAINT'),
   'ua850'      : dict(project='ref_climatos', product='ERAINT'),
   'ua700'      : dict(project='ref_climatos', product='ERAINT'),
   'ua500'      : dict(project='ref_climatos', product='ERAINT'),
   'ua200'      : dict(project='ref_climatos', product='ERAINT'),
   'va850'      : dict(project='ref_climatos', product='ERAINT'),
   'va700'      : dict(project='ref_climatos', product='ERAINT'),
   'va500'      : dict(project='ref_climatos', product='ERAINT'),
   'va200'      : dict(project='ref_climatos', product='ERAINT'),
   'ta850'      : dict(project='ref_climatos', product='ERAINT'),
   'ta700'      : dict(project='ref_climatos', product='ERAINT'),
   'ta500'      : dict(project='ref_climatos', product='ERAINT'),
   'ta200'      : dict(project='ref_climatos', product='ERAINT'),
}
```

2.1.4 Difference with a simulation

Use a simulation dataset as the reference

Atlas Explorer also allows to do differences between a reference simulation and the 'models' datasets.

For this, you simply provide a simulation dataset dictionary to reference:

```
# -- Set the reference against which we plot the diagnostics
# ------------------------------------------------------------------------ >
# --    -> 'default' uses variable2reference to point to a default
# --       reference dataset (obs and reanalyses)
# --    -> you can set reference to a dictionary that will point any other
# --       climaf dataset
# --       For instance, you can set it to models[0] if you want to see the
# --       differences relative to the first simulation of the list 'models'
#reference = 'default'

reference = dict(project = 'CMIP5', model='CNRM-CM5', experiment='historical',
                 frequency='monthly', period='1980-2005',
                 customname='CMIP5 CNRM-CM5'
                 )
```

2.1.4 Difference with a simulation

Use a simulation dataset as the reference

Note 1

CliMAF offers a set of pre-defined plot parameters, depending on the 'context': full_field (for a climatology), 'bias', and 'model_model' for model-model differences. The C-ESM-EP automatically defines the context by analysing the input datasets.

In that case, CliMAF will use the pre-defined plot parameters for the 'model_model' differences (and not the 'bias' differences).

(more details in section 2.2.4)

Note 2

Much of our climate science consists in doing climatologies and differences…

This functionality provides a huge amount of possibilities:

- Differences of targeted periods of a simulation relative to the climatology of this simulation
- Evolution of a simulation relative to the first month/year
- Differences of a set of simulations with a reference model version
- …

2.2 AtlasExplorer parameter file

2.2 Atlas Explorer parameter file

Using the parameter file to control the diagnostics

The content of the Atlas Explorer (and the C-ESM-EP components) is widely controlled from the parameter file:

```
AtlasExplorer/params_AtlasExplorer.py
```

In this section we will learn how to use it.

2.2.1 Overall parameters

General control of the Atlas Explorer

The following parameters control the execution:

```
# -- Set the verbosity of CliMAF (minimum is 'critical',
maximum is 'debug', intermediate -> 'warning')
verbose='debug'

# -- Safe Mode (set to False and verbose='debug' if you want
to debug)
safe_mode = False

# -- Set to 'True' (string) to clean the CliMAF cache
clean_cache = 'False'
```

2.2.1 Overall parameters

General control of the Atlas Explorer

Those parameters are general values for the diagnostics of the atlas:

```
# -- Set the reference against which we plot the diagnostics
# ------------------------------------------------------------------------ >
# --    -> 'default' uses variable2reference to point to a default
# --       reference dataset (obs and reanalyses)
# --    -> you can set reference to a dictionary that will point any other
# --       climaf dataset
# --       For instance, you can set it to models[0] if you want to see the
# --       differences relative to the first simulation of the list 'models'
reference = 'default'


# -- Set the overall season, region and geographical domain
# --> season, region and domain do not overwrite the values that are pre-defined with
some diagnostics
# ------------------------------------------------------------------------ >
season = 'ANM'  # -> Choose among all the possible values taken by clim_average (see
help(clim_average)) like JFM, December,...
proj = 'GLOB' # -> Set to a value taken by the argument 'proj' of plot(): GLOB, NH,
SH, NH20, SH30...
#domain = dict(lonmin=0, lonmax=360, latmin=-30, latmax=30) # -> set domain =
dict(lonmin=X1, lonmax=X2, latmin=Y1, latmax=Y2)
domain = {}
```

2.2.2 atlas_explorer_variables

Provide a variable with additionnal features

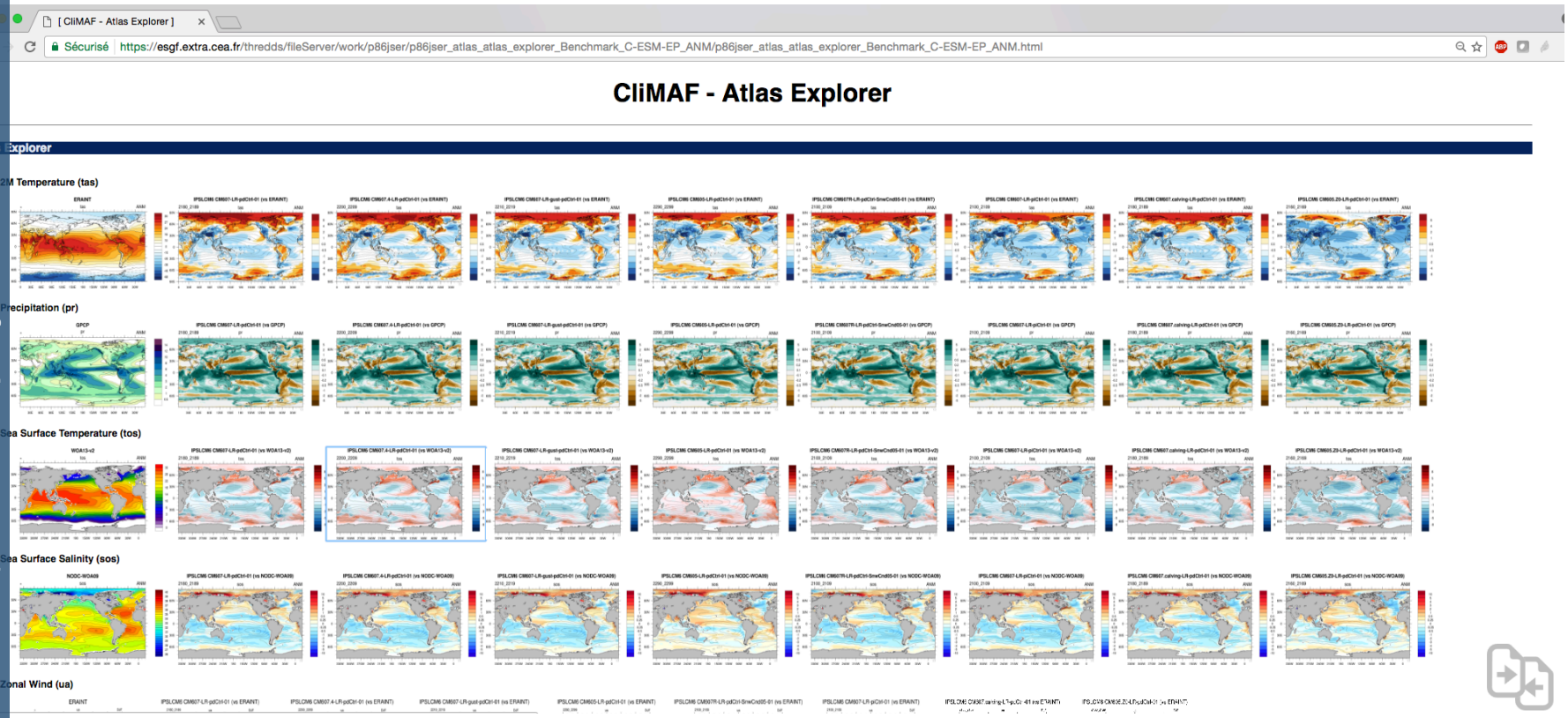
One of the goals of Atlas Explorer is to provide the user with an easy control of the scientific content of the atlas.

Each line of the html page is an element of the atlas_explorer_variables list in the params_AtlasExplorer.py file.

2.2.2 atlas_explorer_variables

Provide a variable with additionnal features

Variables (seasons, projections, domains…)



2.2.2 atlas_explorer_variables

Provide a variable with additionnal features

From the atlas_explorer_variables python list, the user can provide:

- a variable name as a character string
  => uses the overall plot parameters + the pre-defined plot parameters for each variable

- A dictionary containing specific instructions for this line of plot

```
atlas_explorer_variables = ['tas','pr',
                            'tos','sos',
                            dict(variable='ua', season='DJF', add_climato_contours=True),
                            dict(variable='ua', season='JJA', add_climato_contours=True),
                            dict(variable='tos',domain=dict(lonmin=-80,lonmax=40,latmin=10,latmax=85)),
                            dict(variable='sic', proj='NH50', season='March'),
                            dict(variable='lai', season='MAM'),
                           ]
```

2.2.2 atlas_explorer_variables

Provide a variable with additionnal features

Here is the list of arguments that you can control from the dictionary of a variable:

All arguments that the 'plot' CliMAF function can take:

- http://climaf.readthedocs.io/en/latest/scripts/plot.html
- **`proj:`** the projection used (GLOB, NH20, SH60…)
- **`contours:`** add contours
- **`colors, min, max, delta:`** control the color isolines
- **`color:`** an NCL color palette
- …

See next slide

2.2.2 atlas_explorer_variables

Provide a variable with additionnal features

And arguments that are specific to the C-ESM-EP:

- **`season:`** explicit (see clim_average in CliMAF doc)
- **`domain:`** a geographical domain (example in params_AtlasExplorer.py)
- **`spatial_anomalies:`** True or False (remove the mean of the field)
- **`cdogrid:`** provide a specific CDO grid to regrid both model and reference
- **`regrid_option:`** a specific CDO regridding method
- **`zonmean_variable:`** specify that the variable will be shown as zonal means (for new variables created from the param file)
- **`add_climato_contours:`** True/False

2.2.3 The variables

How can I add my own new variable?

Now I would like to work with a new variable (that is not already used in the various parameter files of all the different components).

Case 1

If path/filename targeted by the CliMAF project you are using contains explicitly the variable name (as for CMIP5), you can access it rightaway. Ex with 'tas':

```
tas_Amon_IPSL-CM5A-LR_historical_r1i1p1_185001-200512.nc
```

Case 2

If not, like in the IGCM_OUT project for the 'SE' files, the file contains multiple variables and the file name contains a specific character string. Example with 'tas' found in the 'histmth' file:

```
CM605-LR-pdCtrl01_SE_2000_2009_1M_histmth.nc
```

Go to next slide

2.2.3 The variables

Add my variable + create an alias to the CMIP name

You then need to tell CliMAF that it will **find the variable 'tas' in the file containing 'histmth' (filenameVar='histmth')** by creating an alias with the CliMAF function calias():

```
calias('IGCM_OUT','tas','t2m', filenameVar='histmth')
```

We encourage the users to use CMIP variable names to facilitate the sharing of atlases (and crossing the different CliMAF projects in the same atlas).
That's why at the same time, **we create an alias to tell CliMAF that the variable 'tas' (second argument) is actually the variable 't2m' (third argument) in the IGCM_OUT project (first argument).**

You can also add a scaling and offset to this alias (CliMAF works with SI units):

```
calias("IGCM_OUT", 'tos',           offset=273.15, filenameVar='grid_T')
calias("IGCM_OUT", 'sic', 'siconc', scale=100,     filenameVar="icemod")
```

2.2.3 Creating a derived variable

Create a variable as the combination of variables

A 'derived variable' is a new variable obtained as the combination of existing variables (adding/substracting two variables, extract a PFT…).

The user can create his/her own derived variables from the parameter file.

Example:

```
derive('*','rstt','minus','rsdt','rsut')
```

… means that for all projects (first argument '*', the variable 'rstt' (second argument) is the substraction (application of the operator 'minus', third argument) between 'rsdt' and 'rsut'.
It is possible to add a scale and offset at that step.

Go to next slide…

2.2.3 Creating a derived variable

Create a variable as the combination of variables

For more details on derive(), please look at:
http://climaf.readthedocs.io/en/latest/functions_data.html?highlight=derive

The user can also **create a new CliMAF operator** if needed (if the operator is not already available, see the documentation), and/or use the truly powerful **ccdo() operator to use any CDO operator**.

For an explicit example, **check the ORCHIDEE parameter file**.

2.2.4 Plot parameters

Controlling the plot parameters per variable

As we've seen, with Atlas Explorer the user can control the plot parameters directly from the variable dictionary in atlas_explorer_variables.

If you want to modify the plot parameters for all your atlases (for each variable), across all components, you can do it in the **custom_plot_params.py** file in the main directory (C-ESM-EP/work).

In this file you have the possibility to provide any argument of the CliMAF plot() function as a plot parameter for any targeted variable and context.

2.2.4 Plot parameters

Controlling the plot parameters per variable

Example:

2. jservon@ciclad-ng:~/Evaluation/CliMAF/climaf_installs/climaf_1.0.3_CESMEP/climaf/plot (ssh)

```
# -*- coding: iso-8859-1 -*-
# Created : S.Sénési - nov 2015
# Adapter : J.Servonnat - april 2016

dict_plot_params = {
       'pr' : {
              'default'  : { 'scale' : 86400. , 'color' : 'precip_11lev' , 'contours' : 1 },
            'full_field'   : {'colors':'0.5 1 2 3 4 6 8 10 12 14'  },
            'bias'        : {'color':'MPL_BrBG','colors': '-5 -2 -1 -0.5 -0.2 -0.1 0.1 0.2 0.5 1 2 5' },
            'model_model' : {'color':'precip_diff_12lev','colors': '-5 -2 -1 -0.5 -0.2 -0.1 0.1 0.2 0.5 1 2 5'},
        },
       'hurs' : {
            'default' : {'focus': 'ocean'},
            'full_field'   : {'colors':'72 74 76 78 80 82 84 86 88 90 92' , 'color' : 'precip_11lev' },
            'bias'        : {'colors': '-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10' , 'color':'precip_diff_12lev' },
            'model_model' : {'colors': '-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10' , 'color':'precip_diff_12lev' },
        },
       'rstt' : {
            'full_field'   : {'colors':'0 20 40 60 80 100 120 140 160 180 200 220 240 260 280 300 320', 'color':'WhiteBlueGreenYellowR
ed' },
            'bias'        : {'colors': '-50 -40 -30 -20 -10 -5 0 5 10 20 30 40 50' , 'color':'BlueWhiteOrangeRed' },
            'model_model' : {'colors': '-50 -40 -30 -20 -10 -5 0 5 10 20 30 40 50'},
        },
       'rlut' : {
```

Note: if the isolines in 'full_field' are provided with 'colors' (defined by hand… yes it's boring), the climatology of the reference can be over-imposed to the bias maps as contours.

2.2.4 Plot parameters

Controlling the plot parameters per variable

There are three 'contexts':

- **'full_field'** (for a climatology)
- **'bias'** for a model-obs difference
- **'model_model'** for model-model differences

The C-ESM-EP automatically determines the context by analysing the input datasets.

If you just want to modify something in the existing plot parameters (the isolines, the color palette…), copy the plot parameters of the variable from the CliMAF install in your custom_plot_params.py file:

```
/home/jservon/Evaluation/CliMAF/climaf_installs/
climaf_1.0.3_CESMEP/climaf/plot
```

The content of the custom_plot_params.py file provides an example for the variable 'pr'.

The compareCompanion

The compareCompanion

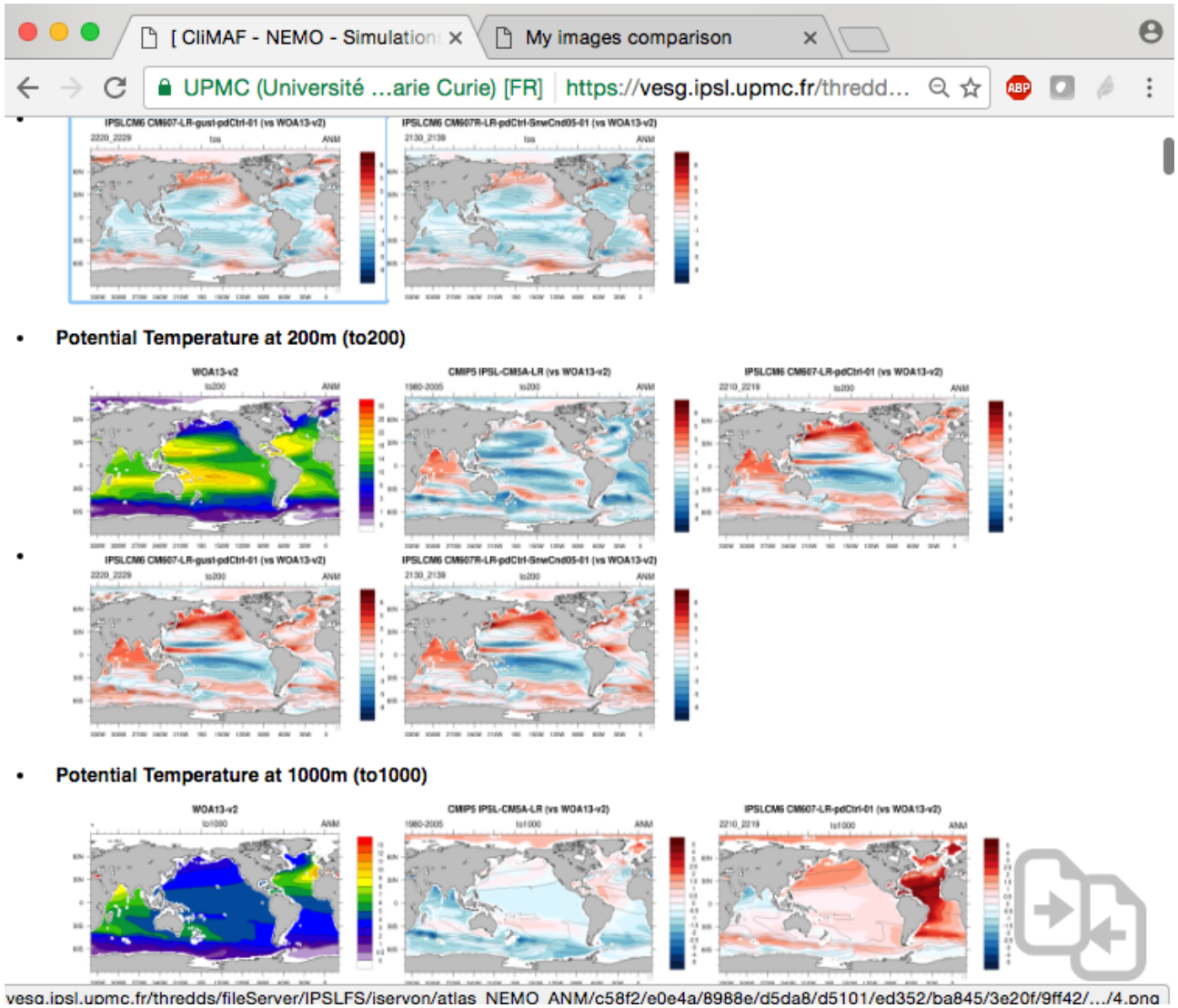
Display a selection of figures on the fly

The various html pages can provide a lot of information and you might need to select a subset of them to focus on a particular question.

For this, the C-ESM-EP comes with an 'on-the-fly' tool to select a subset of plots, called the compareCompanion (P. Brockmann).

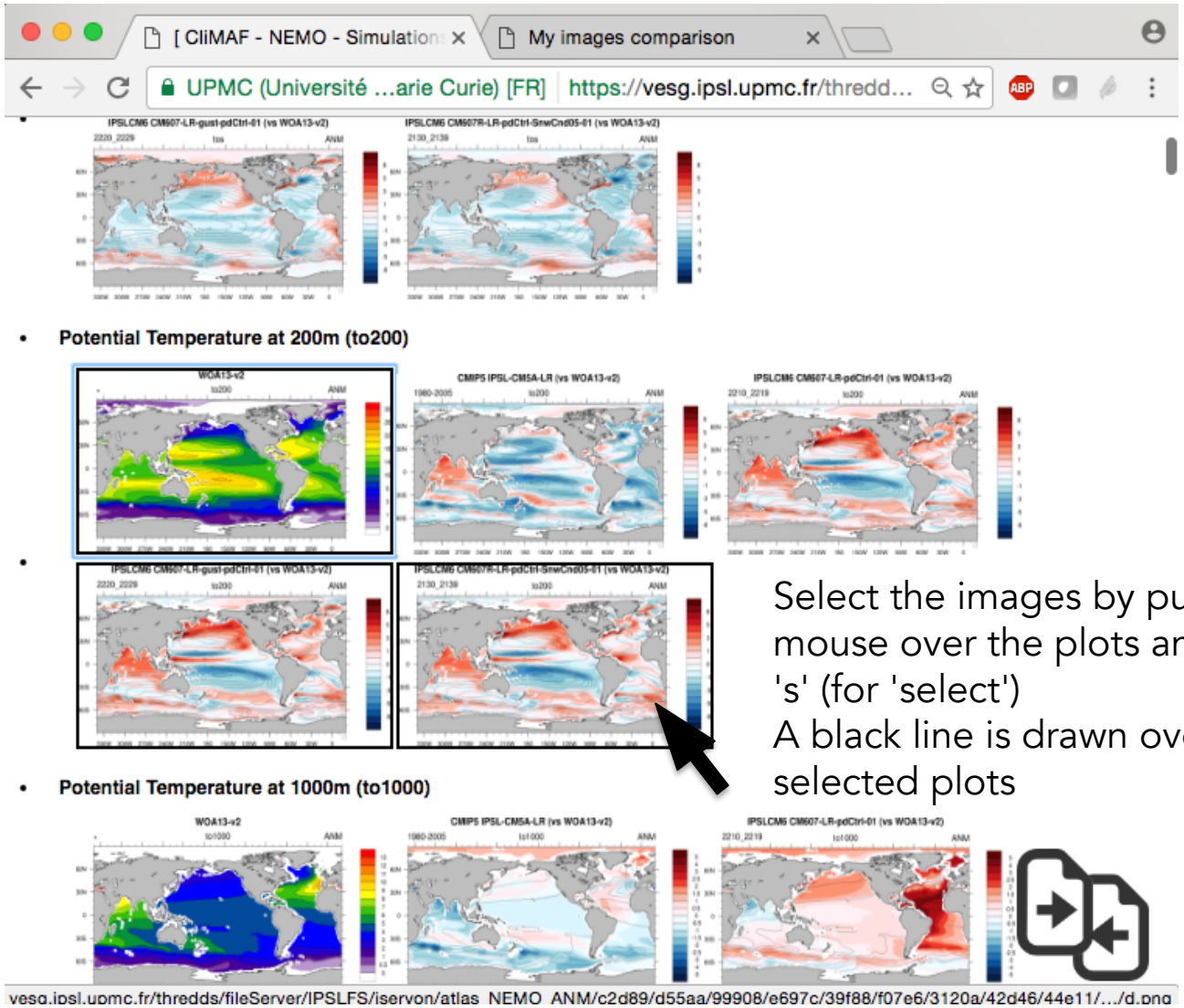
The compareCompanion

Display a selection of figures on the fly



The compareCompanion

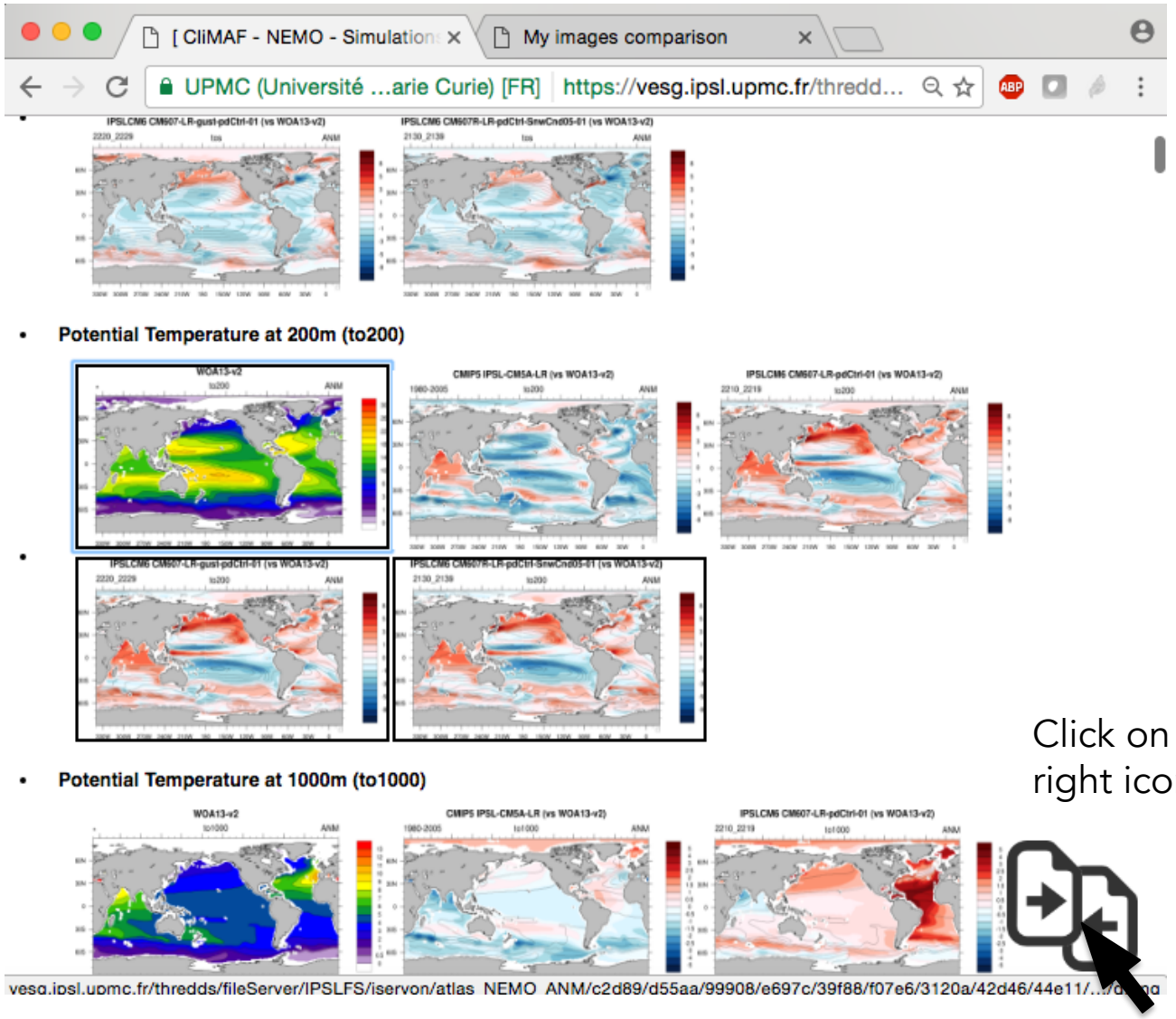
Select your figures with 's'



Select the images by putting the mouse over the plots and press 's' (for 'select')
A black line is drawn over the selected plots

The compareCompanion

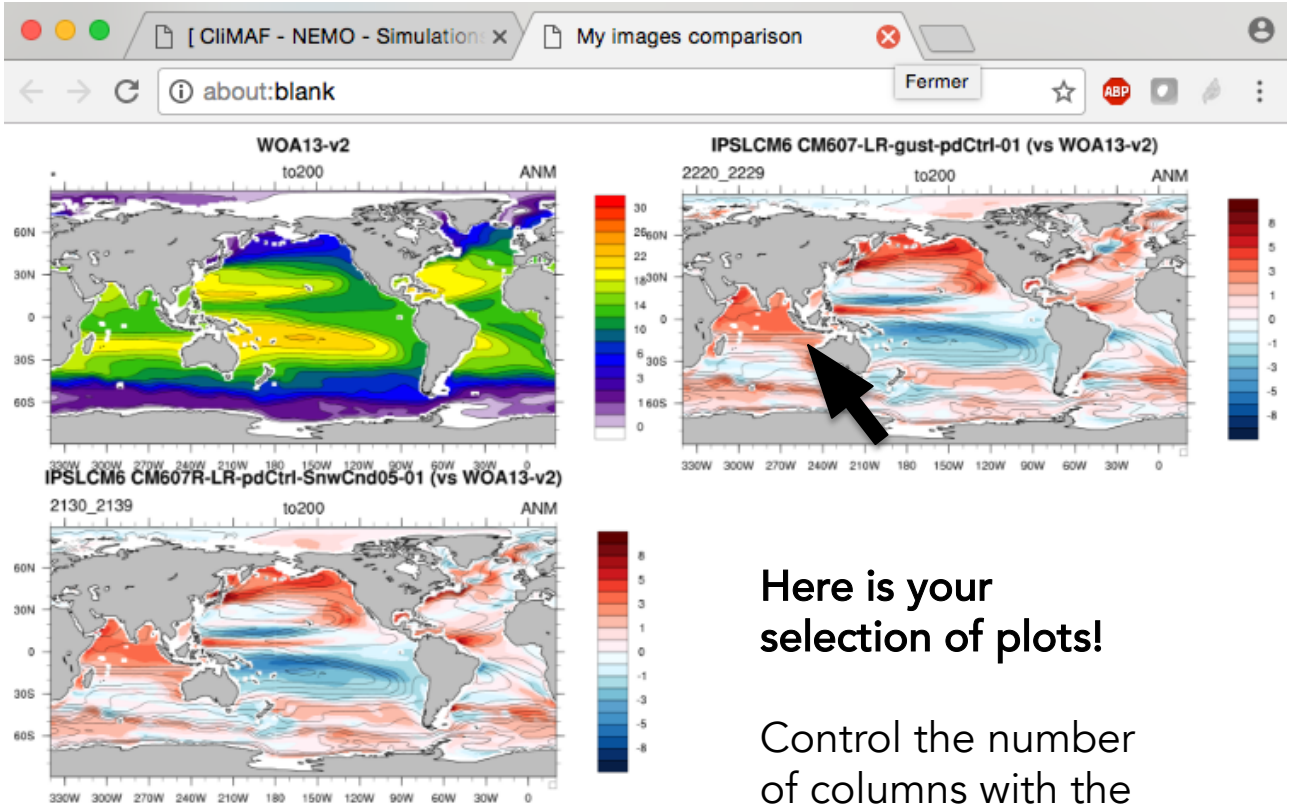
Select your figures with 's'



Click on the bottom right icon…

The compareCompanion

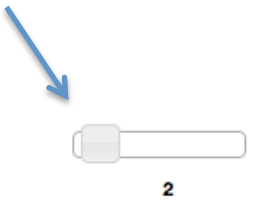
Display a selection of figures on the fly



And switch their positions by dragging the image you want at the position of your choice

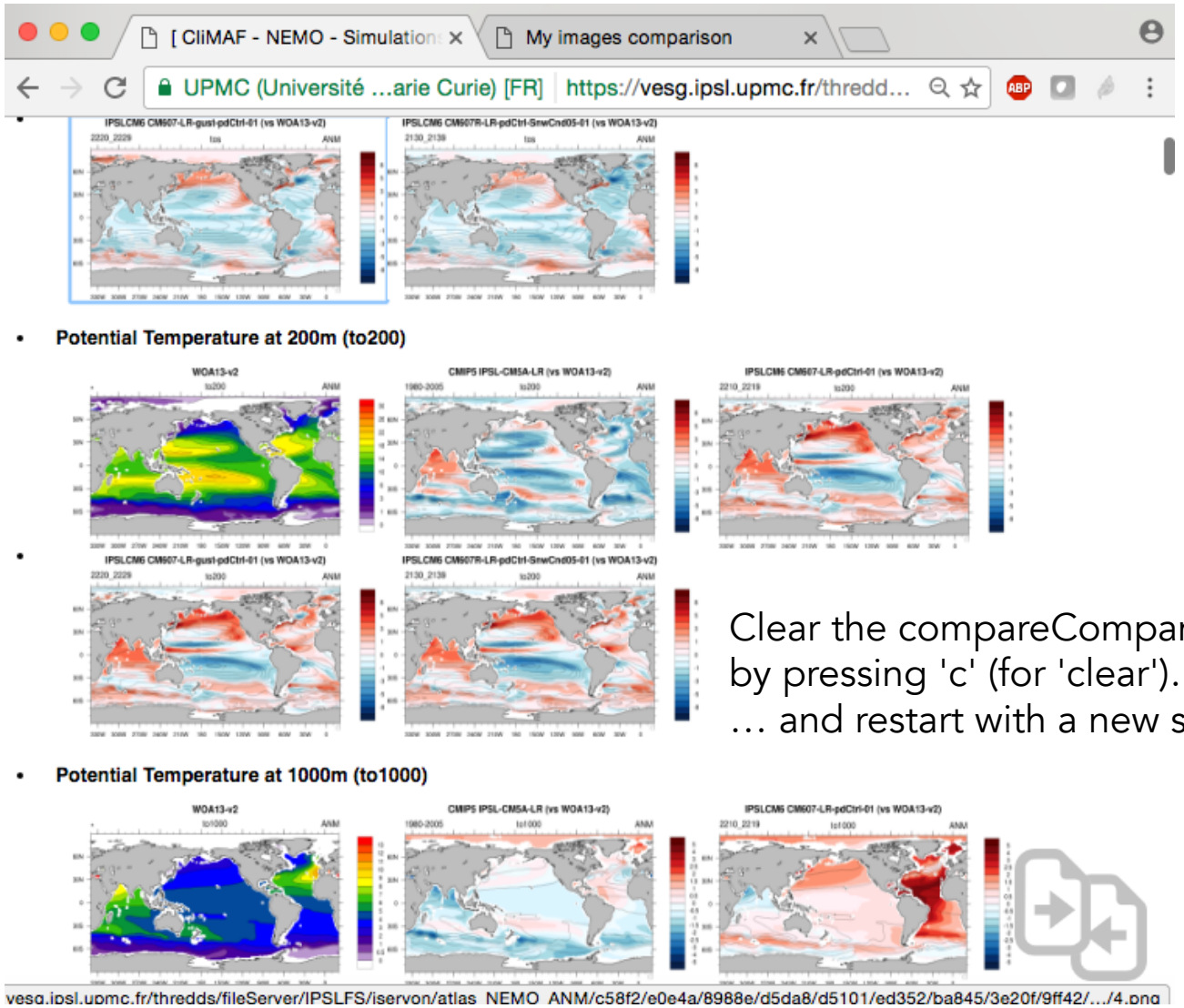
Here is your selection of plots!

Control the number of columns with the slider…



The compareCompanion

Clear the selection



Clear the compareCompanion cache by pressing 'c' (for 'clear')…
… and restart with a new selection.

Questions? jerome.servonnat@lsce.ipsl.fr

CliMAF
Sharing – Simplifying – Optimizing

The CliMAF Earth System Model Evaluation Platform, 2017

J. Servonnat, S. Sénési, L. Vignon, O. Marti, P. Brockmann, S. Denvil

Contributors:

F. Hourdin, I. Musat, M.P. Moine, E. Sanchez, M. Chevallier, R. Msadek, J. Deshayes, M. Van Coppenolle, C. Rousset, J. Mignot, J. Ghattas, P. Peylin, N. Vuichard, P. Cadule, A. Ducharne, F. Maignan, R. Séférian

Beta-testers:

O. Marti, J. Mignot, J. Deshayes, P. Braconnot, P. Sepulchre, M. Kageyama, S. Denvil, R. Séférian, A. Cozic

The authors would like to give special credit to A. Voldoire for showing the way for CliMAF, and to F. Hourdin and the LMDz team for the structure of the C-ESM-EP that is largely inspired from the LMDz evaluation atlas.