clef Documentation

Release 1.3.1.dev3

Scott Wales

Jun 28, 2021

Contents

1 README **3**
- 1.1 Install . . . 3
- 1.2 Use . . . 4

2 Getting Started **9**
- 2.1 Examples . . . 9

3 ESGF command line query **13**

4 Options **15**
- 4.1 clef –missing . . . 15
- 4.2 clef –local . . . 15
- 4.3 Tips . . . 16

5 Climate collections command line query **17**
- 5.1 Examples . . . 18

6 Using CleF queries directly in your code **19**
- 6.1 Examples . . . 19

7 Complex queries **21**
- 7.1 Other examples . . . 22

8 Extra features **25**
- 8.1 CSV file output . . . 25
- 8.2 Query summary option . . . 25
- 8.3 Citations list option . . . 25
- 8.4 Errata and esdoc . . . 26

9 Using CleF - Climate Finder to discover ESGF data at NCI **27**
- 9.1 Command syntax . . . 27
- 9.2 CMIP5 . . . 28

10 CORDEX **31**

11 CMIP6 **33**
- 11.1 Controlling the ouput: clef options . . . 34

12 Requesting new data **37**

13 Integrating the local query in your scripts **39**
13.1 Running search( ) . . . 39

14 AND Filter **43**
14.1 AND filter on more than one attribute . . . 44
14.2 AND filter applied to remote ESGF query . . . 44
14.3 AND filter on the command line . . . 45

15 Other features **47**
15.1 CSV file output . . . 47
15.2 Query summary option . . . 48
15.3 Errata and ESDOC . . . 49

16 More tips on queries **53**
16.1 About experiment_family . . . 53

17 Citation list option **55**

18 Architecture **59**
18.1 clef –missing . . . 59
18.2 clef –local . . . 60

19 CleF API **61**
19.1 clef.db . . . 61
19.2 clef.model . . . 61
19.3 clef.esgf . . . 64

20 Indices and tables **65**

Python Module Index **67**

Index **69**

Contents:

CHAPTER 1

README

Clef searches the Earth System Grid Federation datasets stored at the Australian National Computational Infrastructure, both data published on the NCI ESGF node as well as files that are locally replicated from other ESGF nodes.

Currently it searches for the following datasets:

- **CMIP5** raijin projects: rr3, where NCI is the primary publisher and al33 for replicas
- **CMIP6** raijin projects: 0i10 for replicas
- **CORDEX** raijin projects: rr3, where NCI is the primary publisher and al33 for replicas

The search returns both the path of data that is already available at NCI as well as information on data that is on external ESGF nodes but not yet available locally.

1.1 Install

Clef is pre-installed into a Conda environment at NCI. Load it with:

```
module use /g/data3/hh5/public/modules
module load conda/analysis3-unstable
```

NB You need to be a member of hh5 to load the modules

We are constantly adding new features, the development version is available in a separate environment::
module use /g/data3/hh5/public/modules module load conda source activate clef-test

You can install it to your own environment with:

```
conda install -c coecms -c conda-forge clef
```

But note that the clef.nci.org.au database necessary for running `clef` can only be accessed from NCI systems

1.2 Use

1.2.1 clef cmip5

Find CMIP5 files matching the constraints:

```
clef cmip5 --model BCC-CSM1.1 --variable tas --experiment historical --table day
```

You can filter CMIP5 by the following terms:

- ensemble/member
- experiment
- experiment-family
- model
- table/cmor_table
- realm
- frequency
- variable
- cf-standard-name
- institution

See `clef cmip5 --help` for all available filters and their aliases

`--latest` will check the latest versions of the datasets on the ESGF

website, and will only return matching files

It will return a path for all the files available locally at NCI and a dataset-id for the ones that haven't been downloaded yet.

You can use the flags `--local` and `--missing` to return respectively only the local paths or the missing dataset-id:

```
clef --local cmip5 --model MPI-ESM-LR --variable tas --table day
clef --missing cmip5 --model MPI-ESM-LR --variable tas --table day
```

NB these flags come immediately after the command "clef" and before the sub-command "cmip5" or "cmip6". They are also clearly mutually exclusive. You can repeat arguments more than once:

```
clef --missing cmip5 --model MPI-ESM-LR -v tas -v tasmax -t day -t Amon
```

1.2.2 clef cmip6

You can filter CMIP6 by the following terms:

- activity
- experiment
- source_type
- model
- member

- table
- grid
- resolution
- realm
- frequency
- variable
- version
- sub_experiment
- variant_label
- institution
- cf_standard_name

See `clef cmip6 --help` for all available filters

1.2.3 clef cordex

You can filter CORDEX by the following terms:

- experiment
- domain
- driving_model
- rcm_name (model)
- rcm_version
- ensemble
- table
- time_frequency
- variable
- version
- experiment_family
- institute
- cf_standard_name

See `clef cordex --help` for all available filters

Develop

Development install:

```
conda env create -f conda/dev-environment.yml
source activate clef-dev
pip install -e '.[dev]'
```

The *dev-environment.yml* file is for speeding up installs and installing packages unavailable on pypi, *requirements.txt* is the source of truth for dependencies.

To work on the database tables you may need to start up a test database.

You can start a test database either with Docker:

```
docker-compose up # (In a separate terminal)
psql -h localhost -U postgres -f db/nci.sql
psql -h localhost -U postgres -f db/tables.sql
# ... do testing
docker-compose rm
```

Or with Vagrant:

```
vagrant up
# ... do testing
vagrant destroy
```

Run tests with py.test (they will default to using the test database):

```
py.test
```

or connect to the production database with:

```
py.test --db=postgresql://clef.nci.org.au/postgres
```

Build the documentation using Sphinx:

```
python setup.py build_sphinx
firefox docs/_build/index.html
```

New releases are packaged and uploaded to anaconda.org by CircleCI when a new Github release is made

Documentation is available on ReadTheDocs, both for stable and latest versions.

Disclaimer

CleF can only return datasets which are listed in the ESGF database system for remote results and on the NCI clef database for

- One or more of the ESGF nodes are offline: this can affect clef returning results for the models which are hostedworks which are offline. It is usually easy to verify if this is the case since a query on the browser should show a reduced list of models. In such cases using the *–local* flag will use a query method completely independent and will return at least what is available locally.
- The NCI ESGF node is offline then nothing will be returned by the default or remote queries, again using *–local* should work.
- The checksums stored in the ESGF database are different from the actual file checksums. CleF uses the checksums to match the files available remotely if even one file does not match it will flag the dataset as missing. Using the *–local* flag should still return the datasets regardless because it doesn't compare them to what is available remotely.
- A dataset has been recently donwloaded (up to a week before) and hasn't yet been added to the NCI clef database. In such case it might not show locally even if it has been downloaded. The NCI clef database is updated weekly so we cannot guarantee for clef to find data which is more recent than that. NCI also provides us with a list of datasets recently queued or downloaded. The default query will show this data as

"queued" or "downloaded", rather than missing. While this list aims to cover the gap in between database updates, we have no control on its frequency and it might not capture all the data.

CHAPTER 2

Getting Started

CleF is presently installed in an anaconda environment under the hh5 project. First you will need to get access to this project via https://my.nci.org.au/mancini/project/hh5. Once you have confirmation of your membership, you will need to log out and log back into Gadi for the changes to take effect. If you have issues joining this project, please contact us at cws_help@nci.org.au.

Once you have access to hh5, you must load the anaconda environment before use (on either VDI or Gadi):

```
$ module use /g/data3/hh5/public/modules
$ module load conda/analysis3-unstable
```

NB there is a clef version available on analysis3 but the one in unstable is more recent and has fixes for some bugs.

clef is accessed through the command-line *clef* program. There are presently two main commands:

- `clef cmip5` to execute searches on the CMIP5 dataset
- `clef cmip6` to execute searches on the CMIP6 dataset
- `clef cordex` to execute searches on the CMIP6 dataset
- `clef ds` to execute searches on non-ESGF climate datasets

2.1 Examples

The search works like the ESGF search website, e.g. https://esgf.nci.org.au/search/esgf_nci. Results can be filtered by using flags matching the ESGF search facets.

2.1.1 CMIP5

```
$ clef cmip5 --model ACCESS1.0 \
           --experiment historical \
           --frequency mon \
```

(continues on next page)

(continued from previous page)

```
           --variable ua \
           --variable va

/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r1i1p1/v20120727/ua/
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r1i1p1/v20120727/va/
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r2i1p1/v20130726/ua/
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r2i1p1/v20130726/va/
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r3i1p1/v20140402/ua/
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r3i1p1/v20140402/va/

Everything available on ESGF is also available locally
```

2.1.2 CMIP6

```
$ clef cmip6 --activity CMIP \
           --experiment historical \
           --source_type AOGCM \
           --table Amon \
           --grid gr \
           --resolution "250 km" \
           --variable ua \
           --variable va

/g/data/oi10/replicas/CMIP6/CMIP/CNRM-CERFACS/CNRM-CM6-1/historical/r1i1p1f2/Amon/ua/
↪gr/v20180917/
/g/data/oi10/replicas/CMIP6/CMIP/CNRM-CERFACS/CNRM-CM6-1/historical/r1i1p1f2/Amon/va/
↪gr/v20180917/

Available on ESGF but not locally:
CMIP6.CMIP.CNRM-CERFACS.CNRM-CM6-1.historical.r2i1p1f2.Amon.ua.gr.v20181126
CMIP6.CMIP.CNRM-CERFACS.CNRM-CM6-1.historical.r2i1p1f2.Amon.va.gr.v20181126
```

cordex ++

```
$ clef cordex -dmod CSIRO-BOM-ACCESS1-3 -e historical -v tas -f mon
/g/data/rr3/publications/CORDEX/output/AUS-44/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360J/v1/mon/tas/latest/
/g/data/rr3/publications/CORDEX/output/AUS-44/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360K/v1/mon/tas/latest/
/g/data/rr3/publications/CORDEX/output/AUS-44/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360L/v1/mon/tas/latest/
...
```

2.1.3 ds

```
$ clef ds -f netcdf --standard-name air_temperature
ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_pl/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_pl_<YYYYMMDD>_<YYYYMMDD>.nc
tas: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_sfc/1.0/tas/tas_6hr_ERAI_historical_
↪oper_an_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_ml/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_ml_<YYYYMMDD>_<YYYYMMDD>.nc
mn2t: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/mn2t/mn2t_3hr_ERAI_historical_
↪oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
mx2t: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/mx2t/mx2t_3hr_ERAI_historical_
↪oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
tas: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/tas/tas_3hr_ERAI_historical_
↪oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
```

CHAPTER 3

ESGF command line query

Four optional flags are available for the **cmip5** and **cmip6** commands to change the output or submit a data request:

- `clef --remote <dataset>` returns all the ESGF datasets matching the constraints, it is the equivalent of doing a query online on an ESGF node
- `clef --local <dataset>` finds local files accessing directly the NCI's clef.nci.org.au database so it will also return older versions or datasets that might be temporarily offline.
- `clef --missing <dataset>` finds files on ESGF that haven't been downloaded to NCI
- `clef --request <dataset>` create and pass to NCI a request to download the missing files

If these flags are omitted then the tool will look for the ESGF datasets matching the constraints and return both the local and missing files lists, based on querying an ESGF node.

The query works like the ESGF search website, e.g. https://esgf.nci.org.au/search/esgf_nci. Results can be filtered by using flags matching the ESGF query facets:

```
$ clef cmip5 --model ACCESS1.0 \
          --experiment historical \
          --frequency mon \
          --variable ua \
          --variable va
```

If the same flag is used multiple times both terms will be searched for.

Please note that CMIP5, CMIP6 and CORDEX have different query facets so different names and number of flags in CleF. We tried to use the same names wherever possible. In particular CMIP6 has some new flags available:

```
   $ clef cmip6 --activity CMIP \
             --experiment historical \
             --source_type AOGCM \
             --table Amon \
             --grid gr \
             --resolution "250 km" \
             --variable ua \
```

(continues on next page)

(continued from previous page)

```
              --variable va

`activity` - MIPS or sub-projects, for example CMIP refers to the DECK group of␣
↪experiments
`source_type` - model type, in the example above AOGCM is coupled Atmosphere-Ocean␣
↪Global Climate Model
`grid` - grid kind, in the example 'gr' stands for "regridded data reported on the␣
↪data provider's preferred target grid"
`resolution` - nominal resolution of the grid, there are two kind of nominal␣
↪resolution.
            If the value is in degrees then this is a standard CMIP6 grid, currently␣
↪only "1x1 degree" is available.
            If the resolution is in kms then this is an approximate resolution.␣
↪Details are available in the appendix 2 of the CMIP6 attributes documentation: ␣
↪https://goo.gl/v1drZl
            Note that resolution is always composed of two separate words and will␣
↪need to be passed as a string enclosed in quotes "".
```

While CORDEX has flags specifics to their experiment design:

> *domain* - CORDEX region name *rcm_name* - Identifier of the CORDEX Regional Climate Model *rcm_version* - Identifier for reruns with perturbed parameters or smaller RCM release upgrades *driving_model* - Model/analysis used to drive the model (eg. ECMWFERAINT)

When querying the ESGF website, the total amount of results is limited to 10,000 files. If *clef* finds more results it will ask you to refine your query. You can follow the link to see the query *clef* used on the ESGF website:

```
$ clef cmip5
Exception: Too many results (1030069), try limiting your search
https://esgf.nci.org.au/search/esgf_nci?query=&distrib=on&latest=on&project=CMIP5
```

CHAPTER 4

Options

4.1 clef –missing

`clef --missing <dataset>` queries the ESGF database for files that haven't been downloaded to NCI. It returns ESGF dataset IDs for each dataset that has one or more missing files:

```
$ clef --missing cmip5 --model HadCM3 --experiment historical \
               --table day --ensemble r1i1p1 \
               --variable ta
Available on ESGF but not locally:
cmip5.output2.MOHC.HadCM3.historical.day.atmos.day.r1i1p1.v20110728
```

NOTE: ESGF keeps track of only the most recent versions of each file for a given dataset version, so if the files in the NCI mirror and ESGF don't match this command can return false positives.

4.2 clef –local

`clef --local <dataset>` queries the local file system for files that have been downloaded to NCI. It returns the path to the file on NCI's /g/data disk:

```
$ clef --local cmip5 --model HadCM3 --experiment historical --table day --ensemble␣
↪r1i1p1 \
                    --variable ta --all-versions
/g/data/al33/replicas/CMIP5/combined/MOHC/HadCM3/historical/day/atmos/day/r1i1p1/
↪v20110728/ta
/g/data/al33/replicas/CMIP5/combined/MOHC/HadCM3/historical/day/atmos/day/r1i1p1/
↪v20140110/ta
```

NOTE: Presently the default behaviour for all the ESGF-node based queries is to check for the most recent (latest) version on ESGF, and return only files with that version. This can be disabled with the `--all-versions` flag. The –local option instead currently returns by default all available versions, including versions unpublished by the ESGF

but that are still available locally, Most of the older CMIP5 collection (ua6 project) has been replaced by the new one (al33 project), this does not include older or superceded versions.

4.3 Tips

If your query does not return any results try again at a later time. The tool is querying the ESGF website first and sometimes one or more nodes can be disconnected and the returned results are incomplete. Try the –local flag to at least get what is available locally. For CMIP5 you can use the older ARCCSSive tool if in doubt.

CHAPTER 5

Climate collections command line query

The **ds** command is a new feature of clef and we are still defining its behaviour. clef ds with no other argument will return a list of the local datasets available in the database. NB this is not an exhaustive list of the climate collections at NCI and not all the datasets alredy in the database have been completed.:

```
$ clef ds --help

  Usage: clef ds [OPTIONS]

  Search local database for non-ESGF datasets

  Options:
    -d, --dataset TEXT              Dataset name
    -v, --version TEXT              Dataset version
    -f, --format [netcdf|grib|HDF5|binary]
                                    Dataset file format as defined in clef.db
                                    Dataset table
    -sn, --standard-name [air_temperature|air_pressure|rainfall_rate]
                                   Variable standard_name this is the most
                                   reliable way to look for a variable across
                                   datasets
    -cn, --cmor-name [ps|pres|psl|tas|ta|pr|tos]
                                   Variable cmor_name useful to look for a
                                   variable across datasets
    -va, --variable [T|U|V|Z]      Variable name as defined in files: tas, pr,
                                   sic, T ...
    --frequency [yr|mon|day|6hr|3hr|1hr]
                                   Time frequency on which variable is defined
    --from-date TEXT               To define a time range of availability of a
                                   variable, can be used on its own or
                                   together with to-date. Format is YYYYMMDD
    --to-date TEXT                 To define a time range of availability of a
                                   variable,
                                   can be used on its own or
                                   together with from-date. Format is YYYYMMDD
```

(continues on next page)

(continued from previous page)

```
    --help                          Show this message and exit.
```

shows the available arguments, if you specify any of the variable options then the query will return a list of variables rather then datasets. Since variables can be named differently among datasets, using the standard_nameor cmor_name options to identify them, if available, is the best option.

5.1 Examples

Looking for all datasets which have air temperature data and netcdf as file format:

```
$ clef ds -f netcdf --standard-name air_temperature
  ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_pl/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_pl_<YYYYMMDD>_<YYYYMMDD>.nc
  tas: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_sfc/1.0/tas/tas_6hr_ERAI_historical_
↪oper_an_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
  ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_ml/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_ml_<YYYYMMDD>_<YYYYMMDD>.nc
  mn2t: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/mn2t/mn2t_3hr_ERAI_
↪historical_oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
  mx2t: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/mx2t/mx2t_3hr_ERAI_
↪historical_oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
  tas: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/tas/tas_3hr_ERAI_historical_
↪oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
```

NB for each variable a path structure is returned. This returns a subset of the previous query using the cmor_name to clearly identify one kind of air_temperature:

```
$ clef ds -f netcdf --cmor-name ta
  ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_pl/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_pl_<YYYYMMDD>_<YYYYMMDD>.nc
  ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_ml/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_ml_<YYYYMMDD>_<YYYYMMDD>.nc
```

CHAPTER 6

Using CleF queries directly in your code

The **code** sub-module contains functions which are used to run *–local* option and can be used to integrate this query in your own python scripts:

```
from clef.code import *
```

After importing them you need to open a connection with the NCI local database to be able to run your queries:

```
db = connect()
s = Session()
```

The **search** function takes 4 inputs: the db session, the project (currently 'CMIP5' or 'CMIP6'), latest (True or False) and a dictionary containing the query constraints:

```
df = search(s, project='CMIP5', latest=True, **constraints)
```

The keys available to define your constraints depend on the project you are querying and the attributes stored by the database. You can use any of the *facets* used for ESGF but in future we will be adding other options based on extra fields which are stored as attributes.

6.1 Examples

Here we defined the input dictionary for a CMIP5 query and print out the results dataframe:

```
constraints = {'variable': 'tas', 'model': 'MIROC5', 'cmor_table': 'day', 'experiment
↪': 'rcp85'}
df = search(s, project='CMIP5', **constraints)
df
/g/data/al33/replicas/CMIP5/combined/MIROC/MIRO...   CMIP5  ...  [tas_day_MIROC5_
↪rcp85_r1i1p1_20060101-20091231...
/g/data/al33/replicas/CMIP5/combined/MIROC/MIRO...   CMIP5  ...  [tas_day_MIROC5_
↪rcp85_r2i1p1_20060101-20091231...
/g/data/al33/replicas/CMIP5/combined/MIROC/MIRO...   CMIP5  ...  [tas_day_MIROC5_
↪rcp85_r3i1p1_20060101-20091231...
```

(continues on next page)

(continued from previous page)

```
[5 rows x 12 columns]
'project', 'institute', 'model', 'experiment', 'frequency', 'realm', 'ensemble',
↪'cmor_table', 'version', 'variable', 'path', 'filename'
```

search returns a pandas dataframe, one row for each dataset.

Both the keys and values of the constraints get checked before being passed to the query function. This means that if you passed a key or a value that does not exist for the chosen project, the function will print a list of valid values and then exit. Let's change the constraints dictionary to show an example:

```
constraints = {'v': 'tas', 'm': 'MIROC5', 'table': 'day', 'e': 'rcp85', 'activity':
↪'CMIP'}
df = search(s, project='cmip5', **constraints)
Warning activity is not a valid constraint name
Valid constraints are:
dict_values([['source_id', 'model', 'm'], ['realm'], ['time_frequency', 'frequency',
↪'f'], ['variable_id', 'variable', 'v'], ['experiment_id', 'experiment', 'e'], [
↪'table_id', 'table', 'cmor_table', 't'], ['member_id', 'member', 'ensemble', 'en',
↪'mi'], ['institution_id', 'institution', 'institute'], ['experiment_family']])
```

You can see that the function told us 'activity' is not a valid constraints for CMIP5, in fact that can be used only with CMIP6 We used a shorter version for the keys, we allowed more than one term to be used for each key. The full list is available from the github repository: https://github.com/coecms/clef/blob/master/clef/data/valid_keys.json

More examples and a full description of the function are available in the training page.

CHAPTER 7

Complex queries

We are working on new functions that can facilitate more complex queries, an example is the *matching* function It is easier to understand how matching work starting from an example. We might want to get all the model/ensemble combinations which have both tasmin and tasmax To do this using the standard query we would have to do pass these constraints to the *search()* function:

```
constraints = {'variable': 'tasmin', 'cmor_table': 'day', 'experiment': 'rcp85'}
```

which would select all the model/ensemble which have tasmin / rcp85 / day. Then we would repeat the same for 'tasmax' and finally check which model/ensemble combinations have both. The *matching()* function simplify all of this. First of all, we can pass to it multiple values:

```
constraints = {'variable': ['tasmin','tasmax'], 'cmor_table': ['day'], 'experiment': [
↪'rcp85']}
```

Then we need define the attribute for which we want all the values to be present:

```
allvalues=['variable']
```

We need to define what are the attributes whose combination define a simulation, model and ensemble, i.e. each model/ensemble combination define a simulation, in some cases you might want to add to these also the version:

```
fixed=['model','ensemble']
```

Finally we call *matching*:

```
results, selection = matching(s, allvalues, fixed, **constraints)
```

The function returns two pandas dataframe, in the first *results* each row is a simulation that has either tasmin or tasmax for {rcp85, day}. The second *selection* has only the simulations that has both *tasmin* and *tasmax*:

```
model         ensemble                              ...
ACCESS1.0     r1i1p1    {(tasmax,), (tasmin,)}      ...   (41, 125)
ACCESS1.3     r1i1p1    {(tasmax,), (tasmin,)}      ...   (53, 108)
```

(continues on next page)

(continued from previous page)

```
BCC-CSM1.1     r1i1p1     {(tasmax,), (tasmin,)}  ...  (33, 116)
BCC-CSM1.1(m)  r1i1p1     {(tasmax,), (tasmin,)}  ...  (45, 127)
BNU-ESM        r1i1p1     {(tasmax,), (tasmin,)}  ...  (28, 111)
...                                          ...  ...         ...
[75 rows x 5 columns]
```

7.1 Other examples

Find simulations which have *tasmin* and *tasmax* for both *rcp85* and *rcp45* experiments:

```
constraints = {'variable': ['tasmin','tasmax'], 'cmor_table': ['day'], 'experiment': [
↪'rcp85', 'rcp45']}
allvalues=['variable', 'experiment']
fixed=['model','ensemble']
results, selection = matching(s, allvalues, fixed, **constraints)
```

Find simulations which have *tasmin* and *tasmax* for either *rcp85* or *rcp45* experiments:

```
constraints = {'variable': ['tasmin','tasmax'], 'cmor_table': ['day'], 'experiment': [
↪'rcp85', 'rcp45']}
allvalues=['variable']
fixed=['model','ensemble', 'experiment']
results, selection = matching(s, allvalues, fixed, **constraints)
```

By default we are querying CMIP5 if we want to do the same for CMIP6 we need to change the project value and use the right facet names Find simulations which have *tasmin* and *tasmax* for *piControl* experiment:

```
constraints = {'variable_id': ['tasmin','tasmax'], 'table_id': ['day'], 'experiment_id
↪': ['piControl']}
allvalues=['variable_id']
fixed=['source_id','member_id']
results, selection = matching(s, allvalues, fixed, project='CMIP6', **constraints)
```

In particular for CMIP6, for which data is still getting published, you might want to execute the same query on the remote ESGF data catalogue rather than locally. In that case we change the *local* argument from its default value True to False:

```
constraints = {'variable_id': ['tasmin','tasmax'], 'table_id': ['day'], 'experiment_id
↪': ['piControl']}
allvalues=['variable_id']
fixed=['source_id','member_id']
results, selection = matching(s, allvalues, fixed, project='CMIP6', local=False,␣
↪**constraints)
```

NB currently using the abbreviated version for the constraints keys won't work, you will have to use the attributes full names. As for *search()*, *matching()* also take an additional argument *latest* which is by default True.

You can now call this function also from the command line:

```
clef --local cmip6 -v tasmin -v tasmax  -t Amon -e piControl -mi r1i1p1f1  --and␣
↪variable_id --and experiment_id
AWI-CM-1-1-MR r1i1p1f1 {'v20181218'}
BCC-CSM2-MR r1i1p1f1 {'v20181016'}
...
```

(continues on next page)

(continued from previous page)

```
MIROC6 r1i1p1f1 {'v20181212'}
MRI-ESM2-0 r1i1p1f1 {'v20190222'}
SAM0-UNICON r1i1p1f1 {'v20190323'}
```

You use the *–and* option to pass the *allvalues* arguments, it assumes that *model* and *ensemble* define a simulation.

The CORDEX dataset has more default attributes to define a simulation and they are different depending if you are doing a local or remote query. For local queries:

domain, *driving_model*, *model_id* and *ensemble*

And for remote queries: *domain*, *driving_model*, *rcm_name* and *ensemble*

These differences reflect the fact that the local query uses the file attributes, while the remote query uses the ESGF facets. Unfortunately different terminology have been used.

CHAPTER 8

Extra features

This is an overview of some of the functionalities we added more recently. Examples are included in the training section.

8.1 CSV file output

The *-–csv* option added to the command line will output the query results in a csv file. Rather than getting only the files path, it will list all the available attributes. This currently works with the *–-local* and *–remote* options, it does not yet work for the standard search.

The csv file name will be <project>_query.csv .

8.2 Query summary option

The *-–stats* option added to the command line will print a summary of the query results Currently it prints the following:

- total number of models, followed by their names
- total number of unique model-ensembles/members combinations
- number of models that have N ensembles/members, followed by their names

–stats works when *–local* or *–remote* are specified but not with the default query

8.3 Citations list option

The *–cite* option added to the command line will create a file containing the citations of all the datasets returned by the query. It retrieves the citation information from the DKRZ WDCC server (https://cera-www.dkrz.de/WDCC). This provides citation information only for CMIP6, so this flag is only available with the *cmip6* sub-command. Currently

is only available when running clef with the *–local* or *–remote* flags. The citation lists is saved in a file calledd *cmip_citations.txt* in the working directory.

8.4 Errata and esdoc

There is some work in progress to add functionalities to interact with the ESDOC and the Errata ESGF systems. For the moment these are available only using clef interactively and not via the command line.

CHAPTER 9

Using CleF - Climate Finder to discover ESGF data at NCI

This is a transcript of the clef_demo.ipynb training notebook available on github.

CleF is currently installed in the CMS conda module analysis3, or analysis3-unstable for the latest version. This is managed by the CMS and is available simply by running > module use /g/data3/hh5/public/modules > module load conda/analysis3

You need to be a member of hh5 to use the modules and of one of the CMIP projects: oi10,rr3, fs38, al33 to access the data and the clef database. You could use the module interactively, for the moment we will use its command line options. Let's start!

9.1 Command syntax

```
Usage: clef [OPTIONS] COMMAND [ARGS]...

Options:
  --remote   returns only ESGF search results
  --local    returns only local files matching arguments in local database
  --missing  returns only missing files matching ESGF search
  --request  send NCI request to download missing files matching ESGF search
  --debug    Show debug info
  --help     Show this message and exit.

Commands:
  cmip5   Search ESGF and local database for CMIP5 files Constraints can be...
  cmip6   Search ESGF and local database for CMIP6 files Constraints can be...
  cordex  Search ESGF and local database for CORDEX files.
  ds      Search local database for non-ESGF datasets
```

By simpling running the command **clef** with no arguments, the tool shows the help message and then exits, basically it is equivalent to > clef –help

We can see currently there are 4 sub-commands, **ds** to query non-ESGF collections and one for each ESGF dataset: **cmip5**, **cordex** and **cmip6**. There are also five different options that can be passed before the sub-commands, one we

have already seen is `--help`. The others are used to modify how the tool will deal with the main query output. We will have a look at them and at **ds** later. Let's start from quering some CMIP5 data, to see what we can pass to the **cmip5** sub-command we can simply run it with its `--help` option.

9.2 CMIP5

```
Usage: clef cmip5 [OPTIONS] [QUERY]...

  Search ESGF and local database for CMIP5 files

  Constraints can be specified multiple times, in which case they are
  combined    using OR: -v tas -v tasmin will return anything matching
  variable = 'tas' or variable = 'tasmin'. The --latest flag will check ESGF
  for the latest version available, this is the default behaviour

Options:
  -e, --experiment x              CMIP5 experiment: piControl, rcp85, amip ...
  --experiment_family [Atmos-only|Control|Decadal|ESM|Historical|Idealized|Paleo|RCP]
                                  CMIP5 experiment family: Decadal, RCP ...
  -m, --model x                   CMIP5 model acronym: ACCESS1.3, MIROC5 ...
  -t, --table, --mip [Amon|Omon|OImon|LImon|Lmon|6hrPlev|6hrLev|3hr|Oclim|Oyr|aero|cfOff|c
  -v, --variable x                Variable name as shown in filanames: tas,
                                  pr, sic ...

  -en, --ensemble, --member TEXT  CMIP5 ensemble member: r#i#p#
  --frequency [mon|day|3hr|6hr|fx|yr|monClim|subhr]
  --realm [atmos|ocean|land|landIce|seaIce|aerosol|atmosChem|ocnBgchem]
  --cf_standard_name TEXT         CF variable standard_name, use instead of
                                  variable constraint

  --and [variable|experiment|cmor_table|realm|time_frequency|model|ensemble]
                                  Attributes for which we want to add AND
                                  filter, i.e. --and variable to apply to
                                  variable values

  --institution TEXT              Modelling group institution id: MIROC, IPSL,
                                  MRI ...

  --latest / --all-versions       Return only the latest version or all of
                                  them. Default: --latest

  --replica / --no-replica        Return both original files and replicas.
                                  Default: --no-replica

  --distrib / --no-distrib        Distribute search across all ESGF nodes.
                                  Default: --distrib

  --csv / --no-csv                Send output to csv file including extra
                                  information. Works only with --local and
                                  --remote. Default: --no-csv

  --stats / --no-stats            Write summary of query results. Works only
```

```
                                with --local and --remote. Default: --no-
                                stats

  --debug / --no-debug          Show debug output. Default: --no-debug
  --help                        Show this message and exit.
```

9.2.1 Passing arguments and options

The `--help` shows all the constraints we can pass to the tool, there are also some additional options which can change the way we run our query. For the moment we can ignore these and use their default values. Some of the constraints can be passed using an abbreviation,like `-v` instead of `--variable`. This is handy once you are more familiar with the tool. The same option can have more than one name, for example `--ensemble` can also be passed as `--member`, this is because the terminology has changed between CMIP5 and CMIP6. You can pass how many constraints you want and pass the same constraint more than once. Let's see what happens though if we do not pass any constraint.

```
ERROR: Too many results (3781387), try limiting your search https://esgf.nci.org.au/
↪search/esgf-nci?query=&type=File&distrib=True&replica=False&latest=True&
↪project=CMIP5
```

```
ERROR: No matches found on ESGF, check at https://esgf.nci.org.au/search/esgf-nci?
↪query=&type=File&distrib=True&replica=False&latest=True&project=CMIP5&
↪ensemble=r2i1p1s&experiment=historical&cmor_table=day&variable=tasmin
```

Oops that wasn't reasonable! I mispelled the ensemble "r2i1p1s" does not exists and the tool is telling me it cannot find any matches.

```
Usage: clef cmip5 [OPTIONS] [QUERY]...
Try 'clef cmip5 --help' for help.

Error: Invalid value for '--table' / '--mip' / '-t': invalid choice: days. (choose␣
↪from Amon, Omon, OImon, LImon, Lmon, 6hrPlev, 6hrLev, 3hr, Oclim, Oyr, aero, cfOff,␣
↪cfSites, cfMon, cfDay, cf3hr, day, fx, grids)
```

Made another spelling mistake, in this case the tool knows that I passed a wrong value and lists for me all the available options for the CMOR table. Eventually we are aiming to validate all the arguments we can, although for some it is no possible to pass all the possible values (ensemble for example).

```
/g/data/al33/replicas/CMIP5/combined/CCCma/CanCM4/historical/day/atmos/day/r2i1p1/
↪v20120207/tasmin/
/g/data/al33/replicas/CMIP5/combined/CCCma/CanCM4/historical/day/atmos/day/r2i1p1/
↪v20120612/tasmin/
/g/data/al33/replicas/CMIP5/combined/CCCma/CanESM2/historical/day/atmos/day/r2i1p1/
↪v20120410/tasmin/
....
/g/data/al33/replicas/CMIP5/combined/NOAA-GFDL/GFDL-CM3/historical/day/atmos/day/
↪r2i1p1/v20120227/tasmin/
/g/data/rr3/publications/CMIP5/output1/CSIRO-QCCCE/CSIRO-Mk3-6-0/historical/day/atmos/
↪day/r2i1p1/files/tasmin_20110518/

Everything available on ESGF is also available locally
```

The tool first search on the ESGF for all the files that match the constraints we passed. It then looks for these file locally and if it finds them it returns their path on raijin. For all the files it can't find locally, the tool check an NCI table listing the downloads they are working on. Finally it lists missing datasets which are in the download queue, followed by the datasets that are not available locally and no one has yet requested.

The tool list the datasets paths and dataset_ids, we used to have a `--format file` option but this has been removed in most recent versions.

The query by default returns the latest available version. What if we want to have a look at all the available versions?

```
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r1i1p1/files/clivi_20120115/
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r1i1p1/files/clivi_20120727/
/g/data/rr3/publications/CMIP5/output1/CSIRO-BOM/ACCESS1-0/historical/mon/atmos/Amon/
↪r3i1p1/files/clivi_20140402/

Everything available on ESGF is also available locally
```

The option `--all-versions` is the reverse of `--latest`, which is also the default, so we get a list of all available versions. Since all the ACCESS1.0 data is available on NCI (which is the authoritative source for the ACCESS models) the tool shouldn't find any missing datasets, if it does please let us know about it.

CHAPTER 10

CORDEX

```
Usage: clef cordex [OPTIONS] [QUERY]...

  Search ESGF and local database for CORDEX files.

  Constraints can be specified multiple times, in which case they are
  combined    using OR: -v tas -v tasmin will return anything matching
  variable = 'tas' or variable = 'tasmin'. The --latest flag will check ESGF
  for the latest version available, this is the default behaviour NB. for
  CORDEX data associated to CMIP6 use  the cmip6 command with CORDEX as
  activity_id

Options:
  --latest / --all-versions       Return only the latest version or all of
                                  them. Default: --latest

  --replica / --no-replica        Return both original files and replicas.
                                  Default: --no-replica

  --distrib / --no-distrib        Distribute search across all ESGF nodes.
                                  Default: --distrib

  --csv / --no-csv                Send output to csv file including extra
                                  information. Works only with --local and
                                  --remote. Default: --no-csv

  --stats / --no-stats            Write summary of query results. Works only
                                  with --local and --remote. Default: --no-
                                  stats

  --debug / --no-debug            Show debug output. Default: --no-debug
  -d, --domain FACET              CORDEX region name
  -e, --experiment FACET          Experiment
  -dex, --driving_experiment FACET
                                  CMIP5 experiment of driving GCM or
```

(continues on next page)

(continued from previous page)

```
                                  'evaluation' for re-analysis

  -dmod, --driving_model FACET    Model/analysis used to drive the model (eg.
                                  ECMWFERAINT)

  -m, --rcm_name FACET            Identifier of the CORDEX Regional Climate
                                  Model

  -rcmv, --rcm_version FACET      Identifier for reruns with perturbed
                                  parameters or smaller RCM release upgrades

  -v, --variable FACET            Variable name in file
  -f, --time_frequency FACET      Output frequency indicator
  -en, --ensemble FACET           Ensemble member of the driving GCM
  -vrs, --version FACET           Data publication version
  -cf, --cf_standard_name FACET   CF-Conventions name of the variable
  -ef, --experiment_family FACET  Experiment family: All, Historical, RCP
  -inst, --institute FACET        identifier for the institution that is
                                  responsible for the scientific aspects of
                                  the CORDEX simulation

  --and [domain|experiment|driving_experiment|driving_model|rcm_name|rcm_
↪version|variable|time_frequency|ensemble|version|cf_standard_name|experiment_
↪family|institute]
                                  Attributes for which we want to add AND
                                  filter, i.e. -v tasmin -v tasmax --and
                                  variable will return only model/ensemble
                                  that have both

  --help                          Show this message and exit.
```

cordex works in the same way but some constraints are specific to its experiment design. These are the cordex `domain`, `rcm_name`, `rcm_version` for the regional model, and the `driving_model` and `driving_experiment` for the driving model. CORDEX also does not use tables so you always have to use f`--frequency` to select different timesteps.

```
/g/data/rr3/publications/CORDEX/output/AUS-44/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360J/v1/mon/tas/latest/
/g/data/rr3/publications/CORDEX/output/AUS-44/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360K/v1/mon/tas/latest/
/g/data/rr3/publications/CORDEX/output/AUS-44/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360L/v1/mon/tas/latest/
/g/data/rr3/publications/CORDEX/output/AUS-44i/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360J/v1/mon/tas/latest/
/g/data/rr3/publications/CORDEX/output/AUS-44i/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360K/v1/mon/tas/latest/
/g/data/rr3/publications/CORDEX/output/AUS-44i/UNSW/CSIRO-BOM-ACCESS1-3/historical/
↪r1i1p1/UNSW-WRF360L/v1/mon/tas/latest/

Everything available on ESGF is also available locally
```

CHAPTER 11

CMIP6

```
Usage: clef cmip6 [OPTIONS] [QUERY]...

  Search ESGF and local database for CMIP6 files Constraints can be
  specified multiple times, in which case they are combined using OR:  -v
  tas -v tasmin will return anything matching variable = 'tas' or variable =
  'tasmin'. The --latest flag will check ESGF for the latest version
  available, this is the default behaviour

Options:
  -mip, --activity [AerChemMIP|C4MIP|CDRMIP|CFMIP|CMIP|CORDEX|DAMIP|DCPP|DynVarMIP|FAFMIP|C
  -e, --experiment x              CMIP6 experiment, list of available depends
                                  on activity

  --source_type [AER|AGCM|AOGCM|BGC|CHEM|ISM|LAND|OGCM|RAD|SLAB]
  -t, --table x                   CMIP6 CMOR table: Amon, SIday, Oday ...
  -m, --model, --source_id x      CMIP6 model id: GFDL-AM4, CNRM-CM6-1 ...
  -v, --variable x                CMIP6 variable name as in filenames
  -mi, --member TEXT              CMIP6 member id: <sub-exp-id>-r#i#p#f#
  -g, --grid, --grid_label TEXT   CMIP6 grid label: i.e. gn for the model
                                  native grid

  -nr, --resolution, --nominal_resolution TEXT
                                  Approximate resolution: '250 km', pass in
                                  quotes

  --frequency [1hr|1hrCM|1hrPt|3hr|3hrPt|6hr|6hrPt|day|dec|fx|mon|monC|monPt|subhrPt|yr|yrP
  --realm [aerosol|atmos|atmosChem|land|landIce|ocean|ocnBgchem|seaIce]
  -se, --sub_experiment_id TEXT   Only available for hindcast and forecast
                                  experiments: sYYYY

  -vl, --variant_label TEXT       Indicates a model variant: r#i#p#f#
  --cf_standard_name TEXT         CF variable standard_name, use instead of
```

```
                                  variable constraint

  --and [variable_id|experiment_id|table_id|realm|frequency|member_id|source_
↪id|source_type|activity_id|grid|grid_label|nominal_resolution|sub_
↪experiment_id]
                                  Attributes for which we want to add AND
                                  filter, i.e. --and variable_id to apply to
                                  variable values

  --cite                          Write list of citations for query results,
                                  works only with --remote and --local
                                  options. Default: False

  --institution TEXT              Modelling group institution id: IPSL, NOAA-
                                  GFDL ...

  --latest / --all-versions       Return only the latest version or all of
                                  them. Default: --latest

  --replica / --no-replica        Return both original files and replicas.
                                  Default: --no-replica

  --distrib / --no-distrib        Distribute search across all ESGF nodes.
                                  Default: --distrib

  --csv / --no-csv                Send output to csv file including extra
                                  information. Works only with --local and
                                  --remote. Default: --no-csv

  --stats / --no-stats            Write summary of query results. Works only
                                  with --local and --remote. Default: --no-
                                  stats

  --debug / --no-debug            Show debug output. Default: --no-debug
  --help                          Show this message and exit.
```

The **cmip6** sub-command works in the same way but some constraints are different. As well as changes in terminology CMIP6 has more attributes (*facets*) that can be used to select the data. Examples of these are the **activity** which groups experiments, **resolution** which is an approximation of the actual resolution and **grid**.

11.1 Controlling the ouput: clef options

```
/g/data/oi10/replicas/CMIP6/CMIP/CNRM-CERFACS/CNRM-CM6-1-HR/1pctCO2/r1i1p1f2/Amon/
↪tasmax/gr/v20191021
/g/data/oi10/replicas/CMIP6/CMIP/CNRM-CERFACS/CNRM-CM6-1/1pctCO2/r1i1p1f2/Amon/tasmax/
↪gr/v20180626
/g/data/oi10/replicas/CMIP6/CMIP/CNRM-CERFACS/CNRM-ESM2-1/1pctCO2/r10i1p1f2/Amon/
↪tasmax/gr/v20200529
...
/g/data/oi10/replicas/CMIP6/CMIP/IPSL/IPSL-CM6A-LR/1pctCO2/r1i1p1f1/Amon/tasmin/gr/
↪v20180727
/g/data/oi10/replicas/CMIP6/CMIP/THU/CIESM/1pctCO2/r1i1p1f1/Amon/tasmin/gr/v20200417
```

In this example we used the `--local` option for the main command **clef** to get only the local matching data path as output. Note also that: - we are using abbreviations for the options where available; - we are passing the variable `-v` option twice; - we used the CMIP6 specific option `-g/--grid` to search for all data that is not on the model native grid. This doesn't indicate a grid common to all the CMIP6 output only to the model itself, the same is true for member_id and other attributes.

`--local` is actually executing the query directly on the NCI clef.nci.org.au database, which is different from the default query where the search is executed first on the ESGF and then its results are matched locally. In the example above the final result is exactly the same, whichever way we perform the query. This way of searching can give you more results if a node is offline or if a version have been unpublished from the ESGF but is still available locally.

```
Available on ESGF but not locally:
CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r1i1p1f1.Amon.clw.gr.v20200620
CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r1i1p1f1.Amon.clwvi.gr.v20200620
...
CMIP6.CMIP.THU.CIESM.1pctCO2.r1i1p1f1.Amon.clw.gr.v20200417
CMIP6.CMIP.THU.CIESM.1pctCO2.r1i1p1f1.Amon.clwvi.gr.v20200417
```

This time we used the `--missing` option and the tool returned only the results matching the constraints that are available on the ESGF but not locally (we changed variables to make sure to get some missing data back).

```
CMIP6.CMIP.CNRM-CERFACS.CNRM-CM6-1-HR.1pctCO2.r1i1p1f2.Amon.tasmin.gr.v20191021
CMIP6.CMIP.CNRM-CERFACS.CNRM-CM6-1.1pctCO2.r1i1p1f2.Amon.tasmin.gr.v20180626
...
CMIP6.CMIP.IPSL.IPSL-CM6A-LR.1pctCO2.r1i1p1f1.Amon.tasmin.gr.v20180727
CMIP6.CMIP.NIMS-KMA.KACE-1-0-G.1pctCO2.r1i1p1f1.Amon.tasmin.gr.v20200115
CMIP6.CMIP.THU.CIESM.1pctCO2.r1i1p1f1.Amon.tasmin.gr.v20200417
```

The `--remote` option returns the Dataset_ids of the data matching the constraints, regardless that they are available locally or not.

Please note that `--local`, `--remote` and `--missing` together with `--request`, which we will look at next, are all options of the main command **clef** and they need to come before any sub-commands.

CHAPTER 12

Requesting new data

What should we do if we found out there is some data we are interested to that has not been downloaded or requested yet? This is a complex data collection, NCI, in consultation with the community, decided the best way to manage it was to have one point of reference. Part of this agreement is that NCI will download the files and update the database that **clef** is interrrogating. After consultation with the community a priority list was decided and NCI has started downloading anything that falls into it as soon as become available. Users can then request from the NCI helpdesk, other combinations of variables, experiments etc that do not fall into this list. The list is available from the NCI climate confluence website: Even without consulting the list you can use **clef**, as we demonstrated above, to search for a particular dataset, if it is not queued or downloaded already **clef** will give you an option to request it from NCI. Let's see how it works.

```
%%bash
clef --request cmip6 -e 1pctCO2 -v clw -v clwvi -t Amon -g gr
no
```

```
Available on ESGF but not locally:
CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r1i1p1f1.Amon.clw.gr.v20200620
CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r1i1p1f1.Amon.clwvi.gr.v20200620
...
CMIP6.CMIP.THU.CIESM.1pctCO2.r1i1p1f1.Amon.clw.gr.v20200417
CMIP6.CMIP.THU.CIESM.1pctCO2.r1i1p1f1.Amon.clwvi.gr.v20200417
Do you want to proceed with request for missing files? (N/Y)
 No is default
Your request has been saved in
 /home/581/pxp581/clef/docs/CMIP6_pxp581_20210429T135117.txt
You can use this file to request the data via the NCI helpdesk: help@nci.org.au  or␣
↪https://help.nci.org.au.
```

We run the same query which gave us as a result 4 missing datasets but this time we used the `--request` option after **clef**. The tool will execute the query remotely, then look for matches locally and on the NCI download list. Having found none gives as an option of putting in a request. It will accept any of the following as a positive answer: > Y YES y yes

With anything else or if you don't pass anything it will assume you don't want to put in a request. It still saved the request in a file we can use later.

```
dataset_id=CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r1i1p1f1.Amon.clw.gr.v20200620
dataset_id=CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r1i1p1f1.Amon.clwvi.gr.v20200620
dataset_id=CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r2i1p1f1.Amon.clw.gr.v20200620
dataset_id=CMIP6.CMIP.CAS.FGOALS-f3-L.1pctCO2.r2i1p1f1.Amon.clwvi.gr.v20200620
```

If I answered **yes** the tool would have sent an e-mail to the NCI helpdesk with the text file attached, NCI can pass that file as input to their download tool and queue your request. NB if you are running clef from gadi you cannot send an e-mail so in that case the tool will skip the question and just remind you to send an e-mail to the NCI helpdesk yourself to finalise the request.

CHAPTER 13

Integrating the local query in your scripts

Until now we looked at how to run queries from the command line, but you can use use the same query run by the `--local` option directly in your python code. By doing so you also get access to a lot more information on the datasets returned not only the path. To do so we have first to import some functions from the clef.code sub-module. In particular the **search( )** function and **connect( )** and **Session( )** that we'll use to open a connection to the database.

13.1 Running search( )

search( ) takes 4 inputs: the db session, the project (i.e. currently 'CMIP5', 'CORDEX' or 'CMIP6'), latest (True or False) and a dictionary containing the query constraints: > search(session, project='CMIP5', latest=True, **kwargs)

Let's start by defining some constraints.

The available keys depend on the project you are querying and the attributes stored by the database. You can use any of the *facets* used for ESGF but in future we will be adding other options based on extra fields which are stored as attributes.

The **search( )** function returns a pandas dataframe where every match to the constraints is a row.

Both the keys and values of the constraints get checked before being passed to the query function. This means that if you passed a key or a value that doesn't exist for the chosen project, the function will print a list of valid values and then exit. Let's rewrite the constraints dictionary to show an example.

```
---------------------------------------------------------------------------

ClefException                             Traceback (most recent call last)

<ipython-input-20-c5717342465f> in <module>
      1 constraints = {'v': 'tas', 'm': 'MIROC5', 'table': 'day', 'experiment': 'rcp85
↪', 'activity': 'CMIP'}
----> 2 results = search(s, **constraints)


~/.local/lib/python3.8/site-packages/clef/code.py in search(session, project, latest,␣
↪**kwargs)
```

(continues on next page)

(continued from previous page)

```
     61     valid_keys = get_keys(project)
     62     # check all passed keys are valid
---> 63     args = check_keys(valid_keys, kwargs)
     64     # load dictionary of valid keys for project facets
     65     vocabularies = load_vocabularies(project)


~/.local/lib/python3.8/site-packages/clef/helpers.py in check_keys(valid_keys, kwargs)
    208         facets = [k for k,v in valid_keys.items() if key in v]
    209         if facets==[]:
--> 210             raise ClefException(
    211                 f"Warning {key} is not a valid constraint name"
    212                 f"Valid constraints are:\n{valid_keys.values()}")


ClefException: Warning activity is not a valid constraint nameValid constraints are:
dict_values([['source_id', 'model', 'm'], ['realm'], ['time_frequency', 'frequency',
↪'f'], ['variable_id', 'variable', 'v'], ['experiment_id', 'experiment', 'e'], [
↪'table_id', 'table', 'cmor_table', 't'], ['member_id', 'member', 'ensemble', 'en',
↪'mi'], ['institution_id', 'institution', 'institute'], ['experiment_family'], ['cf_
↪standard_name']])
```

You can see that the function told us `activity` is not a valid constraints for CMIP5, in fact that can be used only with CMIP6 NB. that the search accepted all the other abbreviations, there's a few terms that can be used for each key. The full list of valid keys is available from from the github repository: https://github.com/coecms/clef/blob/master/clef/data/valid_keys.json

```
project                                                      CMIP5
institute                                                    MIROC
model                                                       MIROC5
experiment                                                   rcp85
frequency                                                      day
realm                                                        atmos
ensemble                                                     r1i1p1
cmor_table                                                     day
version                                                   20120710
variable                                                       tas
path             /g/data/al33/replicas/CMIP5/combined/MIROC/MIR...
filename         {tas_day_MIROC5_rcp85_r1i1p1_20900101-20991231...
periods          [(21000101, 21001231), (20800101, 20891231), (...
fdate                                                     20060101
tdate                                                     21001231
time_complete                                                 True
Name: /g/data/al33/replicas/CMIP5/combined/MIROC/MIROC5/rcp85/day/atmos/day/r1i1p1/
↪v20120710/tas, dtype: object
```

NB that `project` is by default 'CMIP5' so it can be omitted when querying CMIP5 data and `latest` is True by default. Set this to *False* if you want to return all the available versions.

13.1.1 Running search() for different sets of attributes

The **search()** function works for one set of attributes, you can specify only one value for each of the attributes at one time. If you want to run a query for two or more different sets of attributes you can call **search()** in a loop. If you have a small numbers of queries then this is easy to implement and run. To make **search()** works for a random number of inputs passed by the command line we set up a function **call_local_query()** that deals with this more efficiently.

The arguments are very similar to **search()** with the important difference that we are passing list of values instead of strings: >call_local_query(s, project, oformat, latest, **kwargs)

Let's look at an example:

Because this function was created to deliver results for the command line local query option, as well as the list of results, it also outputs a list of their paths. Under the hood this function works out all the combinations of the arguments you passed and will run **search()** for each of them, before doing so will also run other functions that check that the values and keys passed to the function are valid. The extra argument `latest` is necessary to resolve the command line `--latest` option.

CHAPTER 14

AND Filter

We started adding additional features to CleF which allows more complex queries. We started from the following case. Let's say that you want to find all the CMIP6 models that have both daily precipitation (pr) and soil moisture (mrso) for a particular experiment(historical). Up to now you would had to select separately both variables and then work out which models had both on your own.

We will show how this work starting by using the actual function interactively. There is also a command line option but it returns only a list of the models. First of all, since we are potentially passing more than one value to the query we are using lists in our *constraints* dictionary. Then we need to define the attributes for which we want all values to be present, only `variable_id` in this case. Finally we tell the function which attributes define a simulation, this would most often be `model` and `member`.

The function returns the selected models/members combinations that have both variables and the corresponding subset of the original query *results*. NB currently using the abbreviated version for the constraints keys won't work, you will have to use the attributes full names. You can see by printing the length of both lists and one of the first item of *selection* that the results have been grouped by models/ensembles and then filtered.

```
174 87
```

```
comb                                              {(pr,), (mrso,)}
frequency                                                    {mon}
version                                                {v20191108}
path         {/g/data/fs38/publications/CMIP6/CMIP/CSIRO-AR...
table_id                                             {Amon, Lmon}
index                                                  (322, 608)
Name: (ACCESS-CM2, r1i1p1f1), dtype: object
```

The full definition the **matching()** shows all the function arguments: >matching(session, cols, fixed, project='CMIP5', local=True, latest=True, **kwargs)

From this you can see that like **search()** by default `project` is 'CMIP5' and `latest` is True. We didn't have to use yet the `local` argument which is True by default, we will see examples later where is set to False so we can do the same query remotely.

14.1 AND filter on more than one attribute

We can pass more than value for more than one attribute, let's add *piControl* to the experiment list.

```
275 93
```

```
comb                                              {(pr,), (mrso,)}
frequency                                                    {mon}
version                                  {v20191112, v20191108}
path         {/g/data/fs38/publications/CMIP6/CMIP/CSIRO-AR...
table_id                                             {Amon, Lmon}
index                                    (322, 624, 680, 766)
Name: (ACCESS-CM2, r1i1p1f1), dtype: object
```

As you can see we get now many more results but only a few more combinations after applying the filter. This is because we are still defining a simulation by using model and member combinations we haven't included experiment and the results for the two experiments are grouped together, to fix this we need to add `experiment_id` to the *fixed* list.

```
270 135
```

```
comb                                              {(pr,), (mrso,)}
frequency                                                    {mon}
version                                               {v20191108}
path         {/g/data/fs38/publications/CMIP6/CMIP/CSIRO-AR...
table_id                                             {Amon, Lmon}
index                                                  (322, 680)
Name: (ACCESS-CM2, r1i1p1f1, historical), dtype: object
```

If we wanted to find all models/members combinations which have both variables and both experiments, then we should have kept *fixed* as it was and add `experiment_id` to the *allvalues* list instead.

```
168 42
```

```
comb         {(mrso, piControl), (mrso, historical), (pr, p...
frequency                                                    {mon}
version                                  {v20191112, v20191108}
path         {/g/data/fs38/publications/CMIP6/CMIP/CSIRO-AR...
table_id                                             {Amon, Lmon}
index                                    (322, 624, 680, 766)
Name: (ACCESS-CM2, r1i1p1f1), dtype: object
```

14.2 AND filter applied to remote ESGF query

You can of course do the same query for CMIP5, in that case you can omit `project` when calling the function since its default value is 'CMIP5'. Another default option is *local=True*, this says the function to perfom this query directly on the local database if you want you can perform the same query on the ESGF database, so you can see what has been published.

```
1494 47
```

```
comb          {(tasmax, historical), (tasmax, rcp26), (tasma...
dataset_id    {cmip5.output1.CNRM-CERFACS.CNRM-CM5.historica...
version              {(v20110629,), (v20110901,), (v20110930,)}
cmor_table                                                {Amon}
index         (422, 423, 424, 425, 426, 427, 476, 477, 478, ...
Name: (CNRM-CM5, r1i1p1), dtype: object
```

Please note how I used different attributes names because we are querying CMIP5 now. `comb` highlights all the combinations that have to be present for a model/ensemble to be returned while we are getting a `dataset_id` rather than a directory path.

14.3 AND filter on the command line

The command line version of **matching** can be called using the `--and` flag followed by the attribute for which we want all values, the flag can be used more than once. By default model/ensemble combinations define a simulation, and only model, ensemble and version are returned as final result.

```
ACCESS1.0 / r1i1p1 versions: 20120727, 20120115
ACCESS1.0 / r2i1p1 versions: 20130726
ACCESS1.0 / r3i1p1 versions: 20140402
...
MRI-CGCM3 / r2i1p1 versions: 20120701
MRI-CGCM3 / r3i1p1 versions: 20120701
MRI-CGCM3 / r4i1p2 versions: 20120701
MRI-CGCM3 / r5i1p2 versions: 20120701
MRI-ESM1 / r1i1p1 versions: 20140210, 20130307
NorESM1-M / r1i1p1 versions: 20120412
NorESM1-M / r2i1p1 versions: 20120412
NorESM1-M / r3i1p1 versions: 20120412
inmcm4 / r1i1p1 versions: 20130207
```

The same will work for `--remote` and *cmip6*

```
ACCESS-CM2 / r1i1p1f1 versions: v20191112
ACCESS-ESM1-5 / r1i1p1f1 versions: v20191214
AWI-ESM-1-1-LR / r1i1p1f1 versions: v20200212
...
NorESM2-MM / r1i1p1f1 versions: v20191108
SAM0-UNICON / r1i1p1f1 versions: v20190910
TaiESM1 / r1i1p1f1 versions: v20200302, v20200211
```

CHAPTER 15

Other features

15.1 CSV file output

The `--csv` option added to the command line will output the query results in a csv file. Rather than getting only the files path, it will list all the available attributes. This currently works only with the `--local` and `--remote` option, it doesn't yet work for the standard search, which is basically a combination of the two.

```
/g/data/al33/replicas/CMIP5/combined/BCC/bcc-csm1-1-m/piControl/mon/atmos/Amon/r1i1p1/
↪v20120705/pr
/g/data/al33/replicas/CMIP5/combined/BCC/bcc-csm1-1/piControl/mon/atmos/Amon/r1i1p1/
↪v1/pr
...
/g/data/rr3/publications/CMIP5/output1/CSIRO-QCCCE/CSIRO-Mk3-6-0/piControl/mon/atmos/
↪Amon/r1i1p1/files/tas_20110518
/g/data/rr3/publications/PMIP3/output/UNSW/CSIRO-Mk3L-1-2/piControl/mon/atmos/Amon/
↪r1i1p1/files/tas_20170728
/g/data/rr3/publications/PMIP3/output/UNSW/CSIRO-Mk3L-1-2/piControl/mon/atmos/Amon/
↪r1i1p1/v20170728/tas
Saving to CMIP5_query.csv
```

```
,model,experiment,frequency,ensemble,cmor_table,version,variable,path,fdate,tdate,
↪time_complete
0,BCC-CSM1.1(m),piControl,mon,r1i1p1,Amon,20120705,pr,/g/data/al33/replicas/CMIP5/
↪combined/BCC/bcc-csm1-1-m/piControl/mon/atmos/Amon/r1i1p1/v20120705/pr,101,40012,
1,BCC-CSM1.1,piControl,mon,r1i1p1,Amon,1,pr,/g/data/al33/replicas/CMIP5/combined/BCC/
↪bcc-csm1-1/piControl/mon/atmos/Amon/r1i1p1/v1/pr,101,50012,
2,BNU-ESM,piControl,mon,r1i1p1,Amon,20120626,pr,/g/data/al33/replicas/CMIP5/combined/
↪BNU/BNU-ESM/piControl/mon/atmos/Amon/r1i1p1/v20120626/pr,14500101,20081231,True
```

In the first example I am passing `--csv` to a CMIP5 query with the `--local` option set. I am hiding the cell output but the tool still prints the results on the display as well as creating the csv file. The following example demonstrate how `--csv` works also for remote CMIP6 queries and with the flag `--and` which allows for more complex filtering of the data and that we haven't looked at yet.

```
ACCESS-CM2 / r1i1p1f1 versions: v20191112
ACCESS-ESM1-5 / r1i1p1f1 versions: v20191214
AWI-ESM-1-1-LR / r1i1p1f1 versions: v20200212
...
NorESM2-MM / r1i1p1f1 versions: v20191108
SAM0-UNICON / r1i1p1f1 versions: v20190910
TaiESM1 / r1i1p1f1 versions: v20200302, v20200211
Saving to CMIP6_query.csv
```

```
,index,version,activity_id,dataset_id,experiment_id,frequency,grid,grid_label,member_
↪id,nominal_resolution,source_id,source_type,sub_experiment_id,table_id,variable_id,
↪score,comb
0,0,"('v20200211',)",CMIP,CMIP6.CMIP.AS-RCEC.TaiESM1.piControl.r1i1p1f1.Amon.pr.gn.
↪v20200211|esgf.rcec.sinica.edu.tw,piControl,mon,finite-volume grid with 0.9x1.25␣
↪degree lat/lon resolution,gn,r1i1p1f1,100 km,TaiESM1,AOGCM,none,Amon,pr,1.0,"('pr',)
↪"
1,1,"('v20200211',)",CMIP,CMIP6.CMIP.AS-RCEC.TaiESM1.piControl.r1i1p1f1.Amon.pr.gn.
↪v20200211|esgf.rcec.sinica.edu.tw,piControl,mon,finite-volume grid with 0.9x1.25␣
↪degree lat/lon resolution,gn,r1i1p1f1,100 km,TaiESM1,AOGCM,none,Amon,pr,1.0,"('pr',)
↪"
2,2,"('v20200211',)",CMIP,CMIP6.CMIP.AS-RCEC.TaiESM1.piControl.r1i1p1f1.Amon.pr.gn.
↪v20200211|esgf.rcec.sinica.edu.tw,piControl,mon,finite-volume grid with 0.9x1.25␣
↪degree lat/lon resolution,gn,r1i1p1f1,100 km,TaiESM1,AOGCM,none,Amon,pr,1.0,"('pr',)
↪"
```

15.2 Query summary option

The `--stats` option added to the command line will print a summary of the query results It works for both `--local` and `--remote` options, but not with the default query. Currently it prints the following: * total number of models, followed by their names * total number of unique model-ensembles/members combinations * number of models that have N ensembles/members, followed by their names

```
Query summary

42 model/s are available:
ACCESS1.0 ACCESS1.3 BCC-CSM1.1 BCC-CSM1.1(m) BNU-ESM CCSM4 CESM1(BGC) CESM1(CAM5)␣
↪CESM1(WACCM) CMCC-CESM CMCC-CM CMCC-CMS CNRM-CM5 CSIRO-Mk3.6.0 CanESM2 EC-EARTH␣
↪FGOALS-s2 FGOALS_g2 FIO-ESM GFDL-CM3 GFDL-ESM2G GFDL-ESM2M GISS-E2-H GISS-E2-H-CC␣
↪GISS-E2-R GISS-E2-R-CC HadGEM2-AO HadGEM2-CC HadGEM2-ES IPSL-CM5A-LR IPSL-CM5A-MR␣
↪IPSL-CM5B-LR MIROC-ESM MIROC-ESM-CHEM MIROC5 MPI-ESM-LR MPI-ESM-MR MRI-CGCM3 MRI-
↪ESM1 NorESM1-M NorESM1-ME inmcm4

A total of 100 unique model-member combinations are available.

  26 model/s have 1 member/s:

     ACCESS1.0: r1i1p1
     ACCESS1.3: r1i1p1
     BCC-CSM1.1: r1i1p1
     ...
     NorESM1-ME: r1i1p1
     inmcm4: r1i1p1

  7 model/s have 3 member/s:
```

(continues on next page)

(continued from previous page)

```
    CESM1(CAM5): r1i1p1, r2i1p1, r3i1p1
    CESM1(WACCM): r2i1p1, r3i1p1, r4i1p1
    FGOALS-s2: r1i1p1, r2i1p1, r3i1p1
    FIO-ESM: r1i1p1, r2i1p1, r3i1p1
    HadGEM2-CC: r1i1p1, r2i1p1, r3i1p1
    MIROC5: r1i1p1, r2i1p1, r3i1p1
    MPI-ESM-LR: r1i1p1, r2i1p1, r3i1p1

  2 model/s have 4 member/s:

    HadGEM2-ES: r1i1p1, r2i1p1, r3i1p1, r4i1p1
    IPSL-CM5A-LR: r1i1p1, r2i1p1, r3i1p1, r4i1p1

  4 model/s have 5 member/s:

    CNRM-CM5: r10i1p1, r1i1p1, r2i1p1, r4i1p1, r6i1p1
    CanESM2: r1i1p1, r2i1p1, r3i1p1, r4i1p1, r5i1p1
    GISS-E2-H: r1i1p1, r1i1p2, r1i1p3, r2i1p1, r2i1p3
    GISS-E2-R: r1i1p1, r1i1p2, r1i1p3, r2i1p1, r2i1p3

  1 model/s have 6 member/s:

    CCSM4: r1i1p1, r2i1p1, r3i1p1, r4i1p1, r5i1p1, r6i1p1

  1 model/s have 9 member/s:

    EC-EARTH: r11i1p1, r12i1p1, r13i1p1, r1i1p1, r2i1p1, r6i1p1, r7i1p1, r8i1p1,␣
↪r9i1p1

  1 model/s have 10 member/s:

    CSIRO-Mk3.6.0: r10i1p1, r1i1p1, r2i1p1, r3i1p1, r4i1p1, r5i1p1, r6i1p1, r7i1p1,␣
↪r8i1p1, r9i1p1
```

15.3 Errata and ESDOC

Another new features are functions that retrieve errata associated to a file and the documents available in the ESDOC system. We are still working to make these accessible from the command line and also to add tracking_ids to our query outputs. In the meantime you can load them and use them after having retrieve the tracking_id attribute in another way (for example with a simple nc_dump or via xarray if in python). Let's start from the errata:

```
You can view the full report online:
https://errata.es-doc.org/static/view.html?uid=99f28ccc-53b3-68dc-8fb1-f7ca4a2d3393
Title: pr and prc have incorrect values at daily and monthly timescales due to an␣
↪incorrect scaling factor
Status: resolved
Description: Within the conversion from CESM's CAM precipitation units (m s-1) to CMIP
↪'s units of (kg m-2 s-1) an incorrect scaling factor was applied. The conversion␣
↪should have been to multiply CAM's values by 1000 kg m-3. Instead, the values were␣
↪multiplied by 1000 and then divided by 86400, resulting in values that are too␣
↪small.
```

As you can see I've chosen a tracking_id that was associated to some errata. First I use the **errata()** function to retrieve

any associated error_ids and then I print out the result using the **print_error()** function. This first retrieve the message associated to any error_id and then prints it in a human readable form, including the url for the original error report. Let's now have a look at how to retrieve and print some documentation from ESDOC.

```
MIP Era > CMIP6
Institute > MIROC
Canonical Name > --
Name > MIROC6
Type > GCM
Long Name > --
Overview > --
Keywords > --
name > MIROC6
keywords > CCSR-AGCM, SPRINTARS, COCO, MATSIRO, atmosphere, aerosol, sea-ice ocean,␣
↪land surface
overview > MIROC6 is a physical climate model mainly composed of three sub-models:␣
↪atmosphere, land, and sea ice....
```

This time we can use directly one function **get_doc()**. It gets three arguments: * the kind of document, can be model, experiment or mip; * the name of the model, experiment or mip; * project for which I want to retrieve the document, by default this is CMIP6. It will retrieve the document online and print out a summary. It will also return the url for the full document report, shown below.

```
https://api.es-doc.org/2/document/search-name?client=ESDOC-VIEWER-DEMO&encoding=html&
↪project=CMIP6&name=MIROC6&type=CIM.2.SCIENCE.MODEL
```

ESDOC works only for CMIP6 and newer ESGF datasets. The World data Center for Climate (WDCC) website holds documentation for both CMIP6 and CMIP5, the **get_wdcc()** function access these documents. In this case rather than the type of document you have to use the dataset_id to retrieve the information.

```
https://cera-www.dkrz.de/WDCC/ui/cerasearch/solr/select?rows=1&wt=json&q=
entry_name_s:cmip5*output1*MIROC*MIROC5
{"responseHeader":{"status":0,"QTime":4,"params":{"q":"entry_name_
↪s:cmip5*output1*MIROC*MIROC5","rows":"1","wt":"json"}},"response":
↪{"numFound":1,"start":0,"maxScore":1,"docs":[{"geo":["ENVELOPE(-180.00, 180.
↪00, 90.00,-90.00)"],"accuracy_report_s":"not filled","specification_s":"not␣
↪filled","completeness_report_s":"not filled","entry_type_s":"experiment",
↪"qc_institute_s":"MIROC","summary_s":"MIROC data of the MIROC5 model as␣
↪contribution for CMIP5 - Coupled ModelnIntercomparison Project Phase 5␣
↪(https://pcmdi.llnl.gov/mips/cmip5).nExperiment design is described in␣
↪detail innhttps://pcmdi.llnl.gov/mips/cmip5/experiment_design.html and␣
↪the list of outputnvariables and their temporal resolutions are given␣
↪innhttps://pcmdi.llnl.gov/mips/cmip5/datadescription.html . The output␣
↪is stored in netCDFnformat as time series per variable in model grid␣
↪spatial resolution. For more informationnon the Earth System model and the␣
↪simulation please refer to the CIM repository.","general_key_ss":["CMIP5",
↪"IPCC","IPCC-AR5","IPCC-DDC","MIROC5","climate simulation"],"entry_name_
↪s":"cmip5 output1 MIROC MIROC5","textSuggest":["cmip5 output1 MIROC MIROC5",
↪"IPCC-AR5_CMIP5","MIM5","IPCC-AR5_CMIP5 (IPCC Assessment Report 5 and␣
↪Coupled Model Intercomparison Project data sets)"],"title_sort":"cmip5␣
↪output1 MIROC MIROC5","date_range_rdt":"[1960-01 TO 2669-12]","progress_
↪acronym_s":"completely archived","consistency_report_s":"not filled",
↪"additional_infos_ss":["standard_output.pdf","Taylor_CMIP5_design.pdf"],
↪"creation_date_dt":"2012-01-13T14:54:16Z","project_acronym_ss":["IPCC-
↪AR5_CMIP5"],"authors_s":"MIROC","model_s":"MIROC5","id":"2320274","entry_
↪acronym_s":"MIM5","project_name_ss":["IPCC-AR5_CMIP5 (IPCC Assessment␣
```

↪Report 5 and Coupled Model Intercomparison Project data sets)"],"hierarchy_
↪steps_ss":["IPCC-AR5_CMIP5","MIM5"],"access_s":"http://cera-www.dkrz.de/
WDCC/CMIP5/Compact.jsp?acronym=MIM5","hierarchy_ss":["project @ 2 @ IPCC-
↪AR5_CMIP5 @"],"_version_":1698339380635107300,"score":1}]},"spellcheck":
↪{"suggestions":[],"correctlySpelled":true,"collations":[]}}

We are still working to add a function that will give a formatted print of the wdcc documents as for the the ESDOC ones.

CHAPTER 16

More tips on queries

16.1 About experiment_family

`experiment_family` is a facet present only for CMIP5 and CORDEX. It allows you to select all the experiments following in the same category. The correspondent in CMIP6 is `activity`. However, not all experiments belong to a family and searching for both `experiment` and `experiment_family` at the same time can give unexpected results. Let's look at an example, if I want to get all the rcps experiments and historical I might be tempted to pass them as constraints in the same query:

```
ERROR: No matches found on ESGF, check at https://esgf.nci.org.au/search/esgf-nci?
↪query=&type=File&distrib=True&replica=False&latest=True&project=CMIP5&
↪ensemble=r1i1p1&experiment=historical&model=CMCC-CM&cmor_table=Omon&variable=tos&
↪experiment_family=RCP
```

The ESGF query uses an *AND* operator for all the constraints we pass. We couldn't find any matches because both the `experiment` and `experiment_family` constraints have to be satiflied. Similarly if we pass rcp45 as experiment as well as the family RCP we will only get the rcp45 results.

```
/g/data/al33/replicas/CMIP5/combined/CMCC/CMCC-CM/rcp45/mon/ocean/Omon/r1i1p1/
↪v20120518/tos/
/g/data/al33/replicas/CMIP5/combined/CMCC/CMCC-CM/rcp45/mon/ocean/Omon/r1i1p1/
↪v20170725/tos/

Everything available on ESGF is also available locally
```

Finally, it is now possible to use `experiment_family` also in the local search:

```
/g/data/al33/replicas/CMIP5/combined/CMCC/CMCC-CM/rcp45/mon/ocean/Omon/r1i1p1/
↪v20120518/tos
/g/data/al33/replicas/CMIP5/combined/CMCC/CMCC-CM/rcp45/mon/ocean/Omon/r1i1p1/
↪v20170725/tos
/g/data/al33/replicas/CMIP5/combined/CMCC/CMCC-CM/rcp85/mon/ocean/Omon/r1i1p1/
↪v20120528/tos
/g/data/al33/replicas/CMIP5/combined/CMCC/CMCC-CM/rcp85/mon/ocean/Omon/r1i1p1/
↪v20170725/tos
```

(continues on next page)

(continued from previous page)

CHAPTER 17

Citation list option

CleF has functions that retrieve errata associated to a file and the documents available in the ESDOC system. Most of these functionalities are currently working only when you use clef interactively. However, since version 1.2 we added a flag to retrieve citations also using the command line. The `--cite` option added to the command line will create a file containing the citations of all the datasets returned by the query. It retrieves the citation information from the DKRZ WDCC server (https://cera-www.dkrz.de/WDCC). This provides citation information only for CMIP6, so this flag is only available with the **cmip6** sub-command. As for the other flags illustrated above, this option works when you select either `--local` or `--remote` queries.

```
CMIP6.CMIP.AS-RCEC.TaiESM1.historical.r1i1p1f1.day.clt.gn.v20200626
CMIP6.CMIP.AWI.AWI-ESM-1-1-LR.historical.r1i1p1f1.day.clt.gn.v20200212
...
CMIP6.CMIP.NUIST.NESM3.historical.r1i1p1f1.day.clt.gn.v20190812
CMIP6.CMIP.SNU.SAM0-UNICON.historical.r1i1p1f1.day.clt.gn.v20190323
Saving to cmip_citations.txt
```

The citations are listed in a cmip_citations.txt file in the current directory.

```
Lee, Wei-Liang; Liang, Hsin-Chien (2020). AS-RCEC TaiESM1.0 model output prepared for␣
↪CMIP6 CMIP historical. Version v20200626. Earth System Grid Federation. https://doi.
↪org/10.22033/ESGF/CMIP6.9755
Danek, Christopher; Shi, Xiaoxu; Stepanek, Christian; Yang, Hu; Barbi, Dirk; Hegewald,
↪ Jan; Lohmann, Gerrit (2020). AWI AWI-ESM1.1LR model output prepared for CMIP6 CMIP␣
↪historical. Version v20200212. Earth System Grid Federation. https://doi.org/10.
↪22033/ESGF/CMIP6.9328
Wu, Tongwen; Chu, Min; Dong, Min; Fang, Yongjie; Jie, Weihua; Li, Jianglong; Li,␣
↪Weiping; Liu, Qianxia; Shi, Xueli; Xin, Xiaoge; Yan, Jinghui; Zhang, Fang; Zhang,␣
↪Jie; Zhang, Li; Zhang, Yanwu (2018). BCC BCC-CSM2MR model output prepared for CMIP6␣
↪CMIP historical. Version v20181127. Earth System Grid Federation. https://doi.org/
↪10.22033/ESGF/CMIP6.2948
Zhang, Jie; Wu, Tongwen; Shi, Xueli; Zhang, Fang; Li, Jianglong; Chu, Min; Liu,␣
↪Qianxia; Yan, Jinghui; Ma, Qiang; Wei, Min (2018). BCC BCC-ESM1 model output␣
↪prepared for CMIP6 CMIP historical. Version v20181220. Earth System Grid Federation.
↪ https://doi.org/10.22033/ESGF/CMIP6.2949
```

Please note that some of the more recently published datasets might not have citations information available yet on the WDCC server. CleF retrieves the information from different fields and them put them together to form a correct citation in the required format. If some of these fields are not available the citation might be incomplete, so always check the entire file before using it. In particular the part of the citation that includes the doi will be missing.

Let's get back to the command line now and have a look at the third command **ds** This command let you query a separate database that contains information on other climate datasets which are available on raijin.

```
Usage: clef ds [OPTIONS]

  Search local database for non-ESGF datasets

Options:
  -d, --dataset TEXT              Dataset name
  -v, --version TEXT              Dataset version
  -f, --format [netcdf|grib|HDF5|binary]
                                  Dataset file format as defined in clef.db
                                  Dataset table

  -sn, --standard-name [air_temperature|air_pressure|rainfall_rate]
                                  Variable standard_name this is the most
                                  reliable way to look for a variable across
                                  datasets

  -cn, --cmor-name [ps|pres|psl|tas|ta|pr|tos]
                                  Variable cmor_name useful to look for a
                                  variable across datasets

  -va, --variable [T|U|V|Z]       Variable name as defined in files: tas, pr,
                                  sic, T ...

  --frequency [yr|mon|day|6hr|3hr|1hr]
                                  Time frequency on which variable is defined
  --from-date TEXT                To define a time range of availability of a
                                  variable, can be used on its own or together
                                  with to-date. Format is YYYYMMDD

  --to-date TEXT                  To define a time range of availability of a
                                  variable, can be used on its own or together
                                  with from-date. Format is YYYYMMDD

  --help                          Show this message and exit.
```

clef ds
with no other argument will return a list of the local datasets available in the database. NB this is not an exhaustive list of the climate collections at NCI and not all the datasets already in the database have been completed.

```
ERA5 v1.0: /g/data/ub4/era5/netcdf/<stream>/<varname>/<year>/
MACC v1.0: /g/data/ub4/macc/grib/<stream>/
YOTC v1.0: /g/data/rq7/yotc
ERAI v1.0: /g/data/ub4/erai/netcdf/<frequency>/<realm>/<stream>/<version>/<varname>/
OSTIA vNA: /g/data/ua8/ostia
TRMM_3B42 v7: /g/data/ua8/NASA_TRMM/TRMM_L3/TRMM_3B42/<YYYY>/
OISST v2.0: /g/data/ua8/NOAA_OISST/AVHRR/v2-0_modified/
```

(continues on next page)

(continued from previous page)

```
MERRA2 v5.12.4: /g/data/rr7/MERRA2/raw/<streamv1>.<version>/<YYYY>/<MM>/
ERAI v1.0: /g/data/ub4/erai/netcdf/<frequency>/<realm>/<stream>/v01/<varname>/
MACC v1.0: /g/data/ub4/macc/netcdf/<frequency>/<realm>/<stream>/v01/<varname>/
YOTC v1.0: /g/data/rq7/yotc
```

If you specify any of the variable options then the query will return a list of variables rather then datasets. Since variables can be named differently among datasets, using the *standard_name* or *cmor_name* options to identify them is the best option.

```
2T: /g/data/ub4/era5/netcdf/surface/2T/<year>/2T_era5_-90 90 -180 179.75_<YYYYMMDD>_
↪<YYYYMMDD>.nc
T: /g/data/ub4/era5/netcdf/pressure/T/<year>/T_era5_-57 20 78 -140_<YYYYMMDD>_
↪<YYYYMMDD>.nc
2T: /g/data/ub4/era5/netcdf/surface/2T/<year>/2T_era5_-90 90 -180 179.75_<YYYYMMDD>_
↪<YYYYMMDD>.nc
T: /g/data/ub4/era5/netcdf/pressure/T/<year>/T_era5_-57 20 78 -140_<YYYYMMDD>_
↪<YYYYMMDD>.nc
ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_pl/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_pl_<YYYYMMDD>_<YYYYMMDD>.nc
tas: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_sfc/1.0/tas/tas_6hr_ERAI_historical_
↪oper_an_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_ml/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_ml_<YYYYMMDD>_<YYYYMMDD>.nc
mn2t: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/mn2t/mn2t_3hr_ERAI_historical_
↪oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
mx2t: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/mx2t/mx2t_3hr_ERAI_historical_
↪oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
tas: /g/data/ub4/erai/netcdf/3hr/atmos/oper_fc_sfc/1.0/tas/tas_3hr_ERAI_historical_
↪oper_fc_sfc_<YYYYMMDD>_<YYYYMMDD>.nc
```

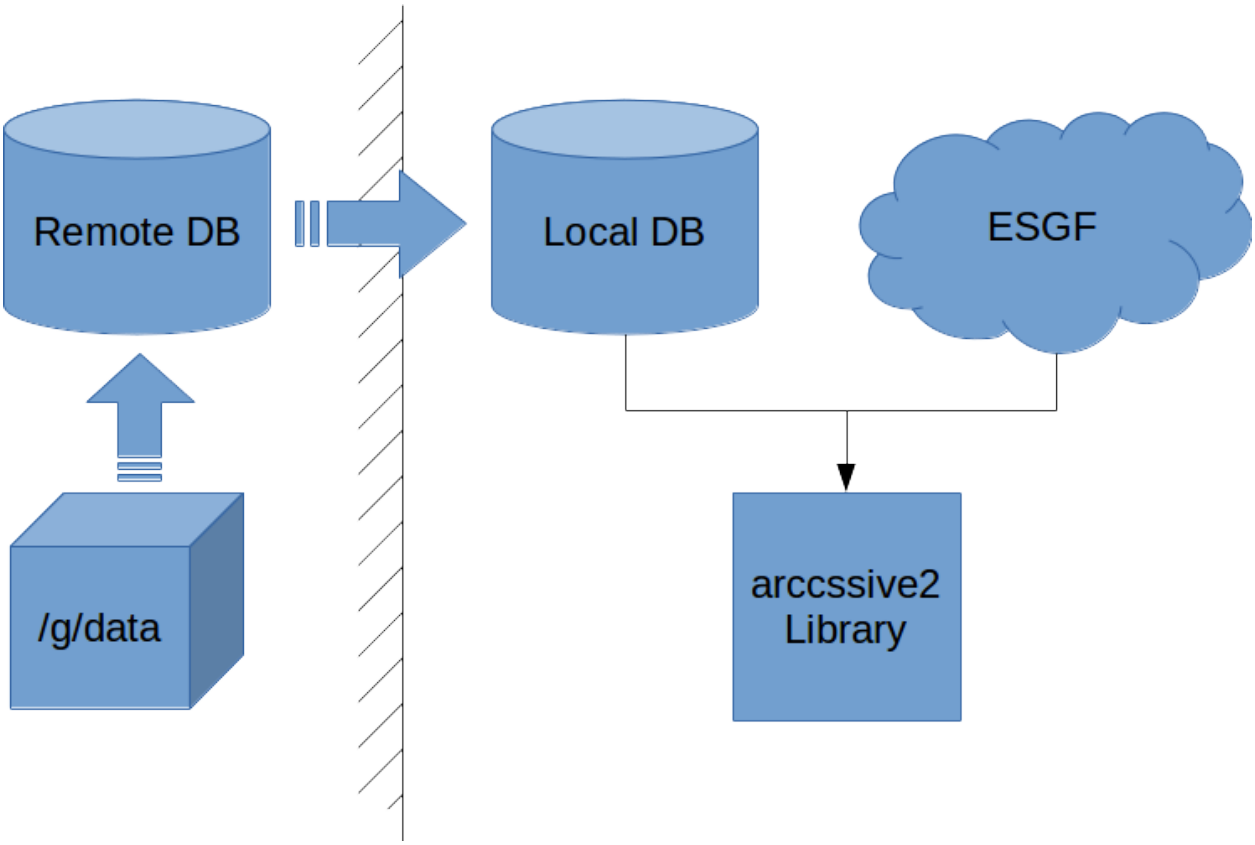
This returns all the variable available as netcdf files and with air_temperature as standard_name. NB for each variable a path structure is returned.

```
T: /g/data/ub4/era5/netcdf/pressure/T/<year>/T_era5_-57 20 78 -140_<YYYYMMDD>_
↪<YYYYMMDD>.nc
T: /g/data/ub4/era5/netcdf/pressure/T/<year>/T_era5_-57 20 78 -140_<YYYYMMDD>_
↪<YYYYMMDD>.nc
ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_pl/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_pl_<YYYYMMDD>_<YYYYMMDD>.nc
ta: /g/data/ub4/erai/netcdf/6hr/atmos/oper_an_ml/1.0/ta/ta_6hr_ERAI_historical_oper_
↪an_ml_<YYYYMMDD>_<YYYYMMDD>.nc
```

This returns a subset of the previous query using the cmor_name to clearly identify one kind of air_temperature.

CHAPTER 18

Architecture



18.1 clef –missing

1. Resolve any constraint wildcards by looking for matches in the local database, e.g.:

```
SELECT DISTINCT model
FROM esgf_dataset
WHERE model ILIKE 'ACCESS%'
;
```

2. Call *find_missing_id()* with the resolved constraints

a. Search ESGF using the constraints, returning the checksum of each matching file

b. Match the ESGF checksums against the local metadata database

c. Return the ESGF id for any files whose checksums cannot be found in the local database

18.2 clef –local

1. Query the local database for files:

```
SELECT path
FROM esgf_paths
NATURAL JOIN esgf_metadata_dataset_link
NATURAL JOIN esgf_dataset
WHERE model ILIKE 'ACCESS%'
-- ...
;
```

2. **If using the `--latest` flag, query ESGF using the constraints to** retrieve checksums, match these checksums against the local results and return only those found. This is the default behaviour

CHAPTER 19

CleF API

19.1 clef.db

Database connection functions

class clef.db.**Session**
sqlalchemy.orm.session.Session connected to the clef.nci.org.au database

connect() must be called before creating any new sessions

clef.db.**connect**(*url='postgresql://clef.nci.org.au:5432/clef'*, *user=None*, *debug=False*)
Connect to the local database and sets up the session

Parameters

- **url** – Database URL
- **user** – Username (password will be prompted via getpass)
- **debug** – Print debugging information

Returns sqlalchemy.engine.Engine

19.2 clef.model

Model of NCI's clef.nci.org.au database

The database has two main tables - path and metadata. These base tables are available in the model as *Path* and *Metadata*, they have a SQLAlchemy relationship so that the two table can be joined in queries.

There may be multiple *Metadata* entries for a single *Path*, these represent different metadata types, such as checksums, netCDF attributes and POSIX file attributes. The type can be identified from *Metadata.type*, and is used as a polymorphic identity to SQLAlchemy's single table inheritance, creating the *Checksum*, *Netcdf* and *Posix* models.

The *C5Dataset* and *C6Dataset* models represent datasets like you would find on ESGF, although without a version. They are created in the database from a `DISTINCT` view of the NetCDF attributes, and can be used to group paths on the filesystem into datasets.

class clef.model.**C5Dataset**(***kwargs*)
A CMIP5-era ESGF dataset

This class only has access to attributes from the file itself, so version information is not present.

See the CMIP documentation for descriptions of the attributes

cmor_table

ensemble

experiment

institute

model

project

realm

time_frequency

class clef.model.**C6Dataset**(***kwargs*)
A CMIP6-era ESGF dataset

This class only has access to attributes from the file itself, so version information is not present.

See the CMIP documentation for descriptions of the attributes

activity_id

experiment_id

frequency

grid_label

institution_id

member_id

nominal_resolution

project

realm

source_id

source_type

sub_experiment_id

table_id

variable_id

variant_label

class clef.model.**Checksum**(***kwargs*)
Checksum of a file on Raijin

md5
md5 checksum

path
type: *Path*

sha256
sha256 checksum

class clef.model.**CordexDataset**(***kwargs*)

class clef.model.**ExtendedMetadata**(***kwargs*)
Extra metadata not present in the file's attributes

class clef.model.**Info**(***kwargs*)
General information about a dataset file

This is a database view, its columns shouldn't be used for searching as they are large and not indexed.

contact

description

further_info_url

license

parent_experiment_id

source

title

tracking_id

variant_info

class clef.model.**Metadata**(***kwargs*)
Generic base class for Metadata of a file on Raijin

See *Posix* and *Netcdf* for specific metadata information

json
Metadata value

path
type: *Path*

type
Metadata type

class clef.model.**Netcdf**(***kwargs*)
NetCDF metadata of a file on Raijin

As would be found by `ncdump -h`

attributes
File attributes

dimensions
File dimensions

variables
File variables

class clef.model.**Path**(***kwargs*)
Path of a file on Raijin, with links to metadata

c5dataset
type: *C5Dataset*

c6dataset
type: *C6Dataset*

checksum
type: *Checksum*

netcdf
type: *Netcdf*

path
File path at NCI

class clef.model.**Posix**(***kwargs*)
Posix metadata of a file on Raijin

As would be found by `ls`

class clef.model.**pg_json_property**(*attr_name*, *index*, *cast_type*)

19.3 clef.esgf

CHAPTER 20

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

clef.db, 61
clef.model, 61

Index

A

activity_id (*clef.model.C6Dataset attribute*), 62
attributes (*clef.model.Netcdf attribute*), 63

C

C5Dataset (*class in clef.model*), 62
c5dataset (*clef.model.Path attribute*), 63
C6Dataset (*class in clef.model*), 62
c6dataset (*clef.model.Path attribute*), 63
Checksum (*class in clef.model*), 62
checksum (*clef.model.Path attribute*), 64
clef.db (*module*), 61
clef.db.Session (*class in clef.db*), 61
clef.model (*module*), 61
cmor_table (*clef.model.C5Dataset attribute*), 62
connect() (*in module clef.db*), 61
contact (*clef.model.Info attribute*), 63
CordexDataset (*class in clef.model*), 63

D

description (*clef.model.Info attribute*), 63
dimensions (*clef.model.Netcdf attribute*), 63

E

ensemble (*clef.model.C5Dataset attribute*), 62
experiment (*clef.model.C5Dataset attribute*), 62
experiment_id (*clef.model.C6Dataset attribute*), 62
ExtendedMetadata (*class in clef.model*), 63

F

frequency (*clef.model.C6Dataset attribute*), 62
further_info_url (*clef.model.Info attribute*), 63

G

grid_label (*clef.model.C6Dataset attribute*), 62

I

Info (*class in clef.model*), 63
institute (*clef.model.C5Dataset attribute*), 62
institution_id (*clef.model.C6Dataset attribute*), 62

J

json (*clef.model.Metadata attribute*), 63

L

license (*clef.model.Info attribute*), 63

M

md5 (*clef.model.Checksum attribute*), 62
member_id (*clef.model.C6Dataset attribute*), 62
Metadata (*class in clef.model*), 63
model (*clef.model.C5Dataset attribute*), 62

N

Netcdf (*class in clef.model*), 63
netcdf (*clef.model.Path attribute*), 64
nominal_resolution (*clef.model.C6Dataset attribute*), 62

P

parent_experiment_id (*clef.model.Info attribute*), 63
Path (*class in clef.model*), 63
path (*clef.model.Checksum attribute*), 62
path (*clef.model.Metadata attribute*), 63
path (*clef.model.Path attribute*), 64
pg_json_property (*class in clef.model*), 64
Posix (*class in clef.model*), 64
project (*clef.model.C5Dataset attribute*), 62
project (*clef.model.C6Dataset attribute*), 62

R

realm (*clef.model.C5Dataset attribute*), 62
realm (*clef.model.C6Dataset attribute*), 62

S

sha256 (*clef.model.Checksum attribute*), 63
source (*clef.model.Info attribute*), 63

source_id (*clef.model.C6Dataset attribute*), 62
source_type (*clef.model.C6Dataset attribute*), 62
sub_experiment_id (*clef.model.C6Dataset attribute*), 62

T

table_id (*clef.model.C6Dataset attribute*), 62
time_frequency (*clef.model.C5Dataset attribute*), 62
title (*clef.model.Info attribute*), 63
tracking_id (*clef.model.Info attribute*), 63
type (*clef.model.Metadata attribute*), 63

V

variable_id (*clef.model.C6Dataset attribute*), 62
variables (*clef.model.Netcdf attribute*), 63
variant_info (*clef.model.Info attribute*), 63
variant_label (*clef.model.C6Dataset attribute*), 62