# Post-processing tools for generating project compliant climate model output

Fabian Wachsmann (wachsmann@dkrz.de)
DKRZ

# Data processing for projects

Idea: Start a science project with a white piece of paper

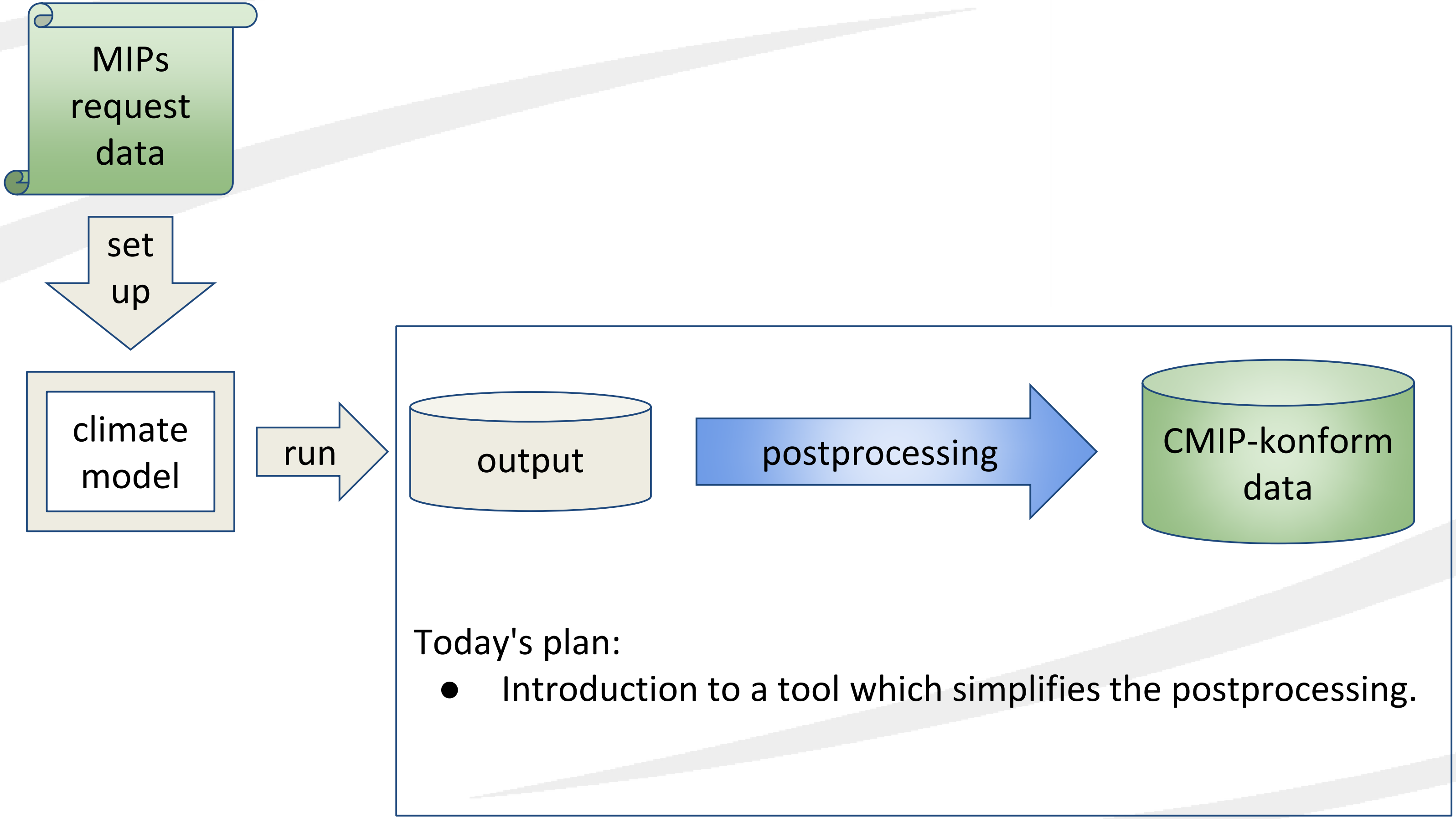
BUT it is not applicable in case of recent model comparison projects because they are growing:

- 2048 variables requested in CMIP6
- Petabyte scale of disk space

How to handle that with limited hpc resources?

We need a plan:
→ Think about a workflow.
→ Use tools.
→ **Do not start from scratch.**

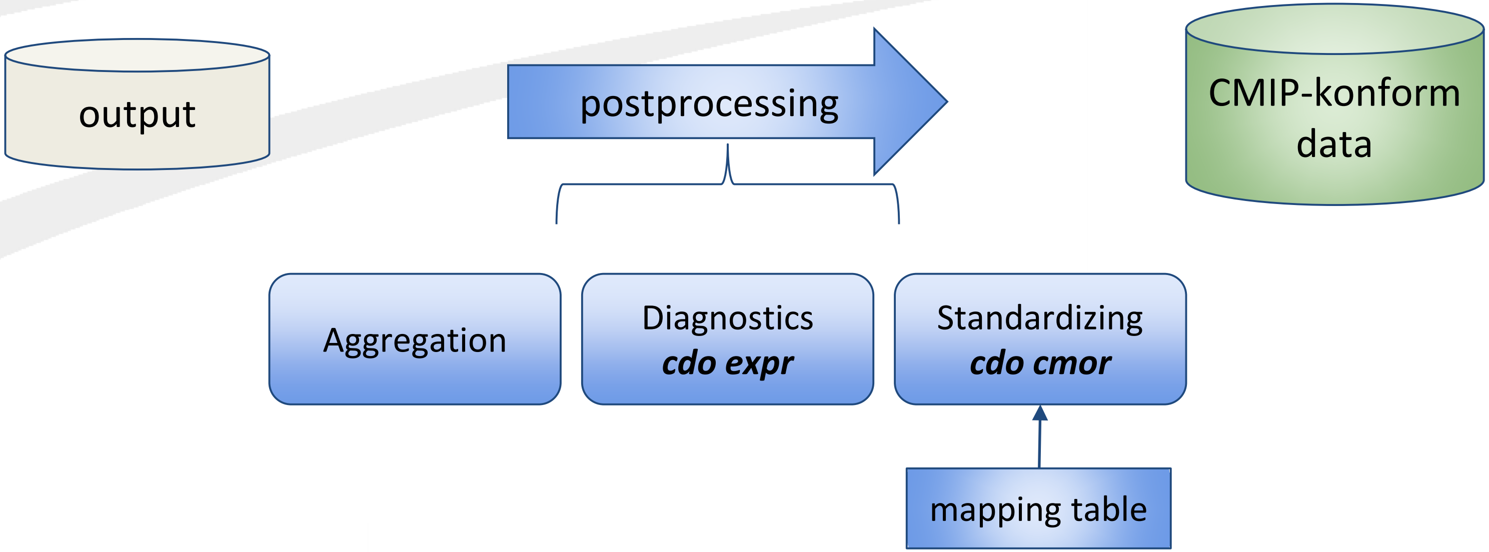
# CMIP/CORDEX data production workflow



Today's plan:

- Introduction to a tool which simplifies the postprocessing.

# Tools

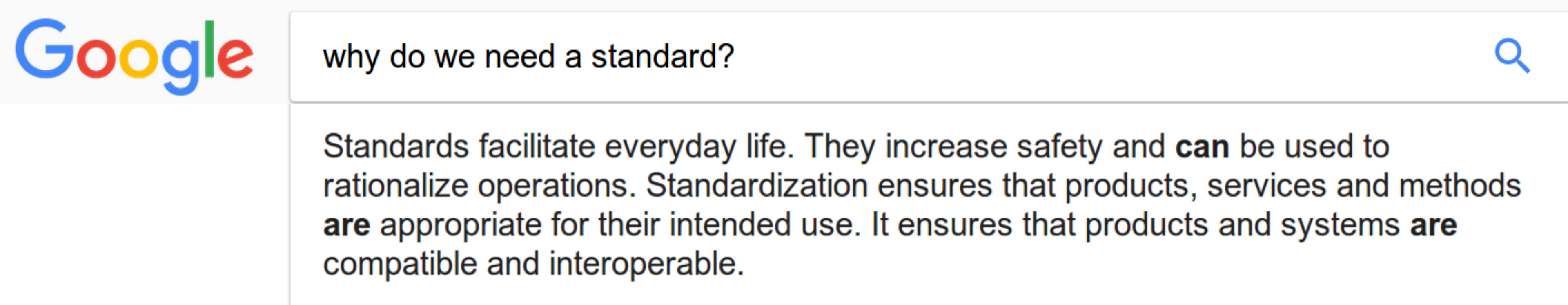


We are developing

- the **cdo cmor** operator
- the **c6dreq-WebGUI** to
  - create a mapping table (map model output variables to CMOR variables)
  - generate Diagnostics and Standardizing script fragments

# Reasons for a data standard

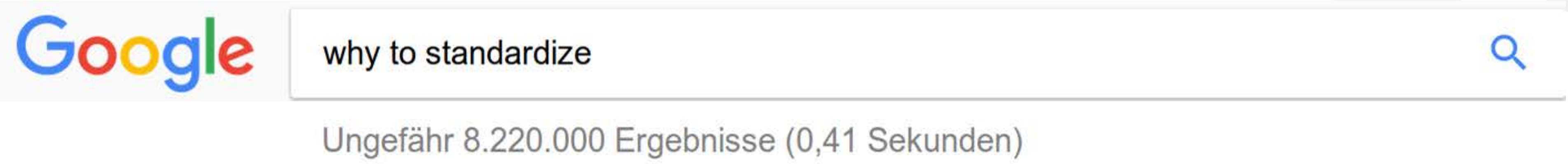
Ask google...



why do we need a standard?

Standards facilitate everyday life. They increase safety and **can** be used to rationalize operations. Standardization ensures that products, services and methods **are** appropriate for their intended use. It ensures that products and systems **are** compatible and interoperable.

But use standard syntax...



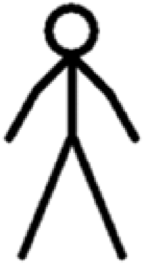
why to standardize

Ungefähr 8.220.000 Ergebnisse (0,41 Sekunden)

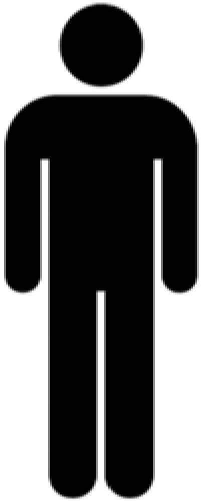
# Complexity of project standards

Scientist who needs to create the standard

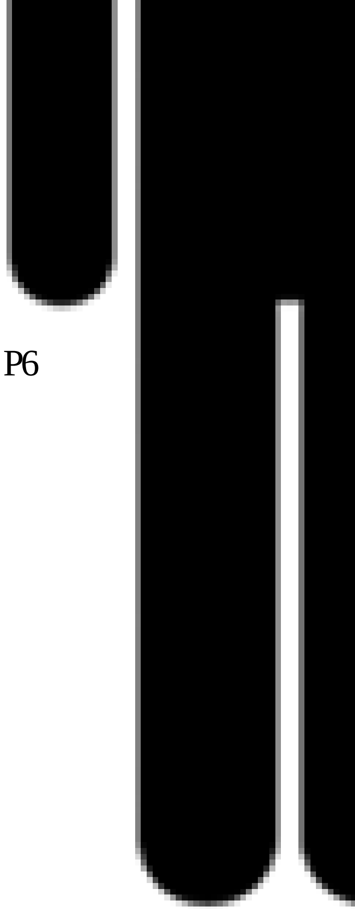
CORDEX Standard



CMIP5 Standard

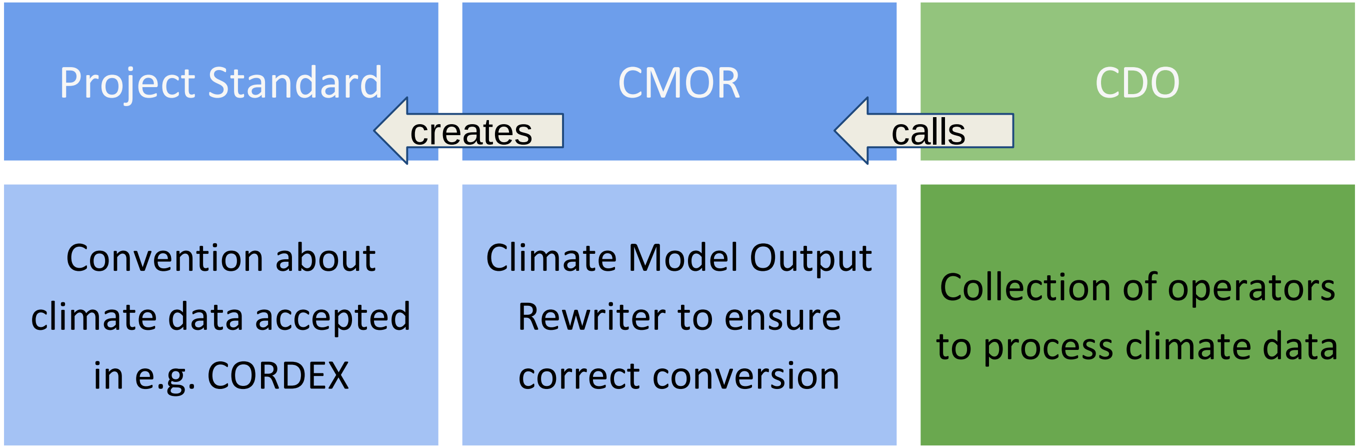


CMIP6



CORDEX2? CMIP7??
→ we need a flexible tool! (CORDEX2?)

# Definitions and motivations



Project Standard

creates

CMOR

calls

CDO

Convention about climate data accepted in e.g. CORDEX

Climate Model Output Rewriter to ensure correct conversion

Collection of operators to process climate data

"Systematic analysis across models only easy to do if model output is written in

- a common format
- with files structured similarly
- and with sufficient metadata uniformly stored“

# CMIP Standard

**General requirements on CMIP compliant data**

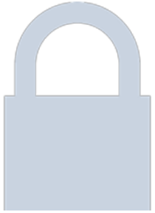
- netCDF4 Format
- conform to CF 1.7
- Each file must contain only a single output field from a single simulation including coordinates and additional meta data

Detailled requirements for the output design of CORDEX can be found in the document cordex_archive_design

More information about global attributes, controlled vocabulary and filenames can be found in the document
CMIP6_global_attributes_filenames_CVs _from Taylor et. al (2017)

# CMOR

- **Why use CMOR to create CMIP standard?**
- Why integrate CDO with CMOR?



- CMOR ensures that output is CMIP compliant.
- Different CMIP standards can be produced
- Use synergies, avoid to repeat work

*No user side preparation of CMIP format description*

# CDO

- Why use CMOR to create CMIP standard?
- **Why integrate CDO with CMOR?**

Use the power of CDOs...

- CDO is a well known tool with an active support
- The CDO's interface allows
  - netCDF as well as GRIB input
  - enables access to all infile information

# cdo cmor

- Why use CMOR to create CMIP standard?
- **Why integrate CDO with CMOR?**

… to simplify the CMOR usage:

```
cmor_setup();
cmor_dataset_json();
cmor_load_table();
cmor_set_table();
cmor_axis();
cmor_grid();
cmor_set_grid_mapping();
cmor_time_varying_grid_coordinate();
cmor_zfactor();
cmor_variable();
cmor_set_deflate();
cmor_set_variable_attribute();
cmor_create_output_path();
cmor_write();
cmor_close();
```

are included by one **cdo cmor** operator

# cdo cmor

**CORDEX_mon**

contains parts of the data request in a CMOR-readable format

**grid_info.nc**

contains a grid description including coordinates and bounds

```
variables:
    double lat(lat);
    double
lat_bnds(lat,bnds);
```

```
cdo cmor,CORDEX_mon,
    gi=grid_info.nc,\
     i=config.txt,\
    mt=mapping_table.txt\
                      infile
```

**config.txt**

contains the user configuration

```
project_id="CORDEX"
```

**mapping_table.txt**

links model output variables with CMOR variables

```
&parameter pmt=mon cmor_name=tas code=201 /
```

# CORDEX_mon

- Header Info:

```
frequency: mon
cmor_version: 2.6! version of CMOR that can read this table
```

- Allowed axes definitions:

```
!=============
axis_entry: height2m
!=============
standard_name:         height
units:                 m
```

- Definitions of requested variables:

```
!============
variable_entry:            ta850
!============
standard_name:             air_temperature
units:                     K
cell_methods:              time: mean
dimensions:                longitude latitude time plev850
```

# grid_info.nc

```
dimensions:
    rlon = 424 ;
    rlat = 412 ;
    vertices = 4 ;
variables:
    double lon(rlat, rlon) ;
           lon:standard_name = "longitude" ;
           lon:bounds = "lon_bnds" ;
    double lon_bnds(rlat, rlon, vertices) ;
    double lat(rlat, rlon) ;
           lat:standard_name = "latitude" ;
           lat:bounds = "lat_bnds" ;
    double lat_bnds(rlat, rlon, vertices) ;
    double rlon(rlon) ;
           rlon:long_name = "longitude in rotated pole grid" ;
    double rlat(rlat) ;
           rlat:long_name = "latitude in rotated pole grid" ;
    int rotated_pole ;
           rotated_pole:grid_north_pole_latitude = 39.25 ;
           rotated_pole:grid_north_pole_longitude = -162. ;
           rotated_pole:grid_mapping_name = 
"rotated_latitude_longitude";
```

# config.txt

```
#control keywords:
mip_table_dir="cordex_mip_tables/"
tracking_prefix=y
keep_all_attributes=y
#required global attributes:
PROJECT_ID=CORDEX
DRIVING_MODEL_ID=CCCma-CanESM2
EXPERIMENT_ID=historical
MEMBER=r1i1p1
CORDEX_DOMAIN="EUR-11"
MODEL_ID=CLMcom-CCLM4-8-17
INSTITUTE_ID="CLMcom"
```

can be created with [https://c6dreq.dkrz.de/cdocmorinfo/index.html](https://c6dreq.dkrz.de/cdocmorinfo/index.html)

# mapping_table.txt

```
&parameter cmor_name=tasmax  pmt=mon code=201 units="K" cell_methods="m" /
&parameter cmor_name=tasmax  pmt=day code=201 units="K" cell_methods="m" /
&parameter cmor_name=rsds    units="W m-2" positive="d" cell_methods="m" /
&parameter cmor_name=rlds   name=rl units="W m-2" positive="d" cell_methods="m"
/
```

- A CMOR-variable is the unique combination of MIP-table (`pmt`) and CMOR name (`cmor_name`).
- CMOR-variables are linked to model variables via `name` (netCDF infile format) or `code` (GRIB infile format).
- Attributes are <u>input</u> attributes! CMOR is able to change
  - `units`
  - and `positive` (positive defined flux direction, can be u[pward] or d[ownward])

  to the requested ones.
- Input `cell_methods` must match the requested `cell_methods`. There are:
  - “m”=mean
  - “p”=point for instantaneous values
  - “c”=climate for 30-year mean
  - “d”=diurnal
  - “n”=none for fixed fields

https://c6dreq.dkrz.de

# Installation

| Instruction\Target format: | **CMIP5/CORDEX** | **CMIP6/CORDEX2** |
|---|---|---|
| *Use DKRZ HPC mistral or install CDO..* | locally via script | locally via script or use conda |
| *Install CDO with...* | CMOR2 | CMOR3 |

# Thanks for your attention!

questions to wachsmann@dkrz.de
find a hands-on on
http://slides.com/wachsylon/cdo-cmor-handson#/

find support on https://c6dreq.dkrz.de
and
https://code.mpimet.mpg.de/projects/cdo