Utrecht University

Dealing with non-CMIP ensembles

Some tips and tricks resulting from ~4.5 years of struggles

Arthur Oldeman – ESM meeting 14 may 2024

What will be treated?

Basically my current workflow for model ensemble data management, processing and standardization (and analysis)

With some critiques, suggestions, and tips

Our aim…

Let’s say you want to investigate **monthly** variability of **atmosphere** and **ocean** interaction on **basin scale** in a **non-CMIP** ensemble of different climate models…

(based on a personal journey)

For example… El Niño and North Pacific atmosphere teleconnections, or… AMOC and the North Atlantic Oscillation, or… Southern ocean variability and surface winds, …

Step 0: think about what we actually need?

I want to analyze:

- 100 years monthly data
- SSTs (ocean) and precipitation (atm)
- Of 2 different simulations
- From 17 different models (PlioMIP2)

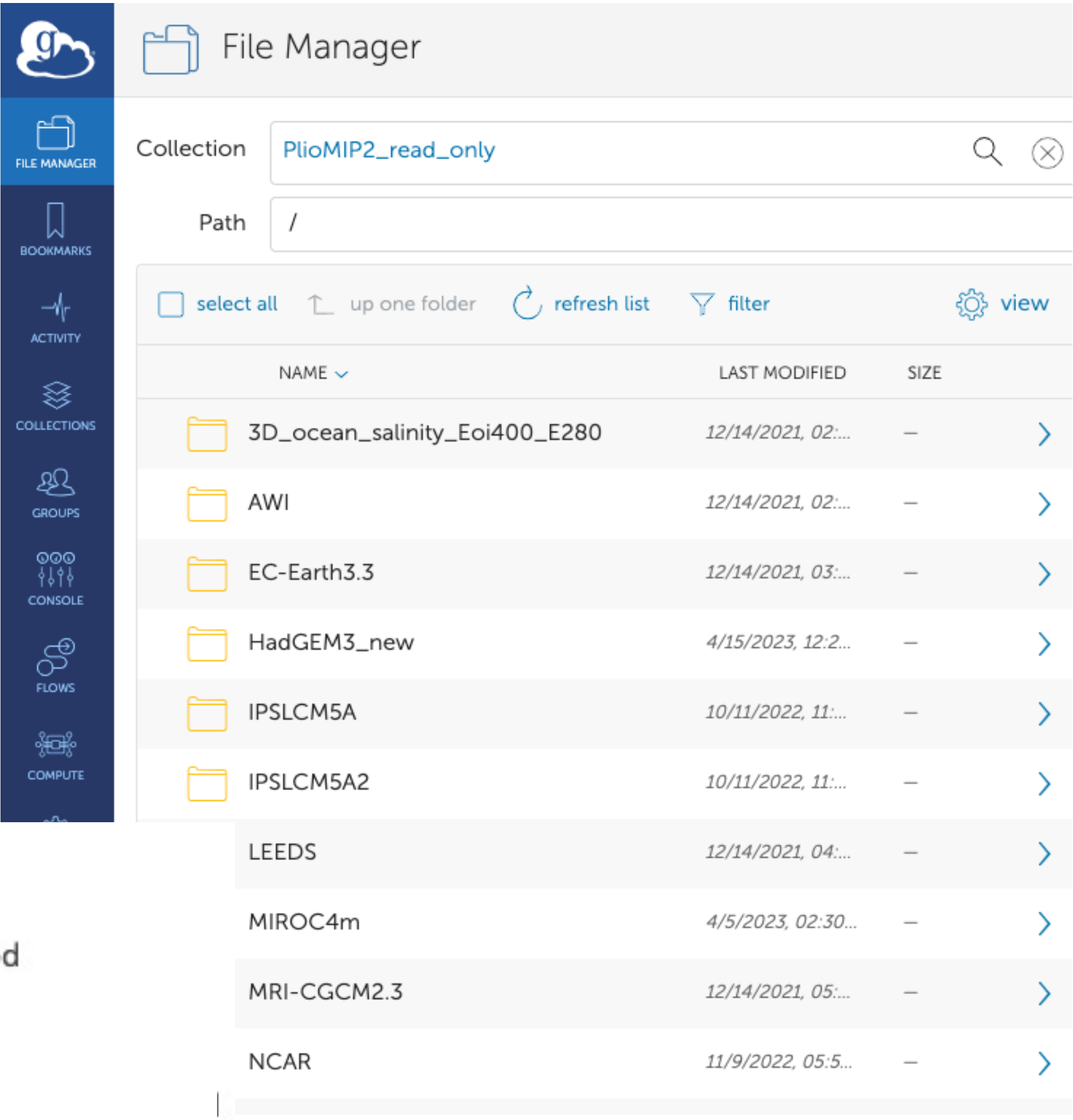
Step 1: Collecting the data

1a. Find the data

Globus server:

access by emailing someone

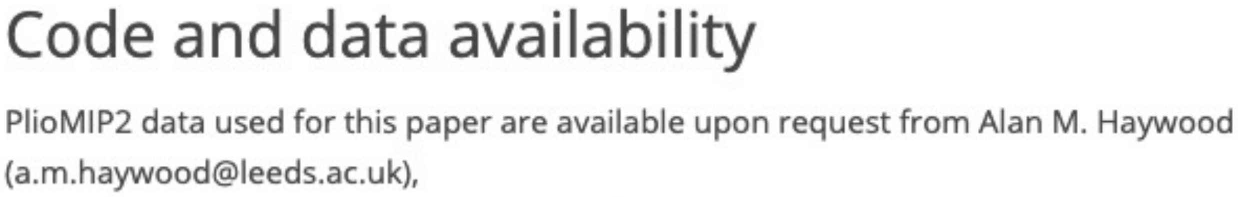
(stated in the “data availability” section of papers..)



Collection PlioMIP2_read_only

Path /

	NAME	LAST MODIFIED	SIZE
	3D_ocean_salinity_Eoi400_E280	12/14/2021, 02:...	–
	AWI	12/14/2021, 02:...	–
	EC-Earth3.3	12/14/2021, 03:...	–
	HadGEM3_new	4/15/2023, 12:2...	–
	IPSLCM5A	10/11/2022, 11:...	–
	IPSLCM5A2	10/11/2022, 11:...	–
	LEEDS	12/14/2021, 04:...	–
	MIROC4m	4/5/2023, 02:30...	–
	MRI-CGCM2.3	12/14/2021, 05:...	–
	NCAR	11/9/2022, 05:5...	–



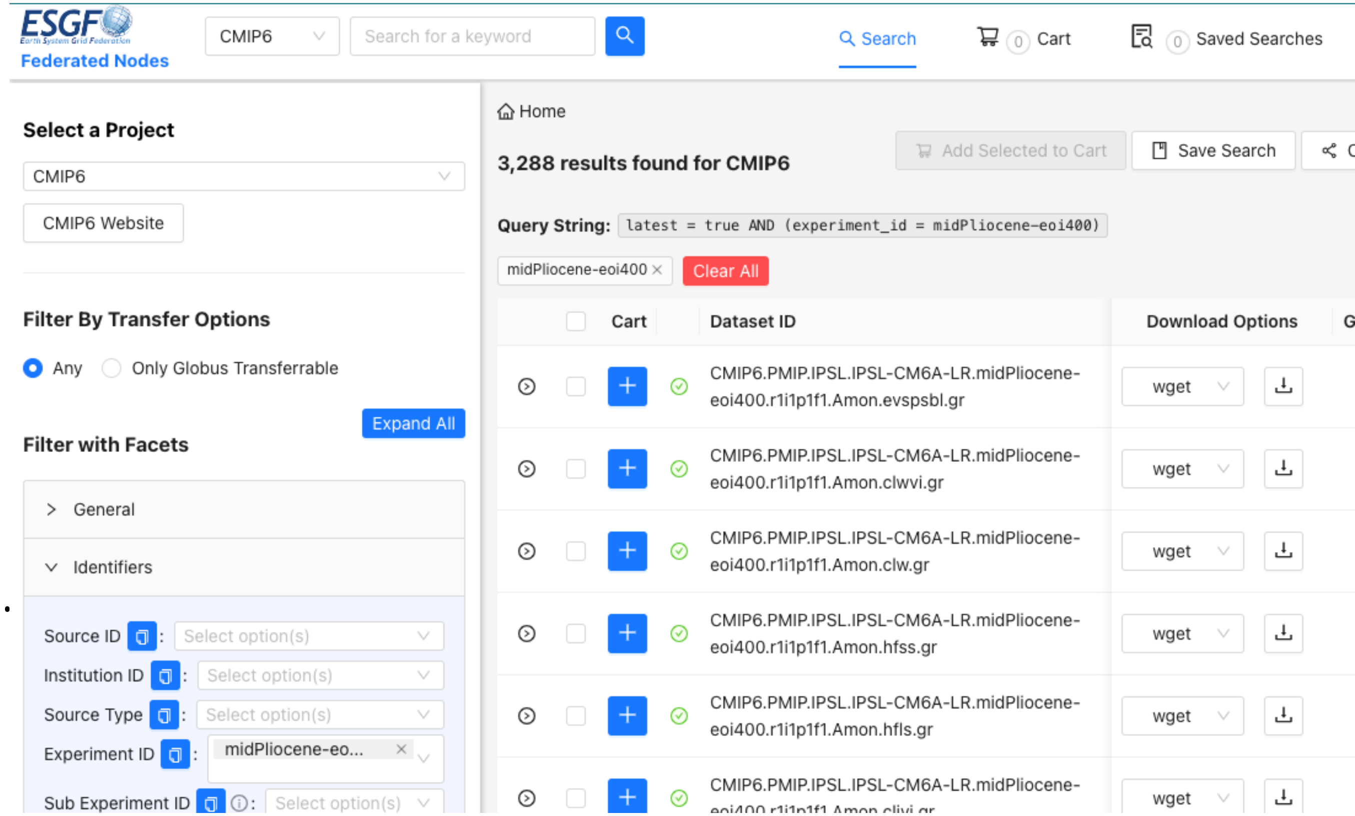
Code and data availability

PlioMIP2 data used for this paper are available upon request from Alan M. Haywood (a.m.haywood@leeds.ac.uk),

Step 1: Collecting the data

1a. Find the data

Some (CMIP6) are on the ESGF grid (can't find if you search for "pliomip" …



ESGF Federated Nodes

CMIP6 | Search for a keyword | Search | 0 Cart | 0 Saved Searches

Home

3,288 results found for CMIP6 | Add Selected to Cart | Save Search

Query String: `latest = true AND (experiment_id = midPliocene-eoi400)`

midPliocene-eoi400 × | Clear All

Select a Project

CMIP6

CMIP6 Website

Filter By Transfer Options

● Any ○ Only Globus Transferrable

Expand All

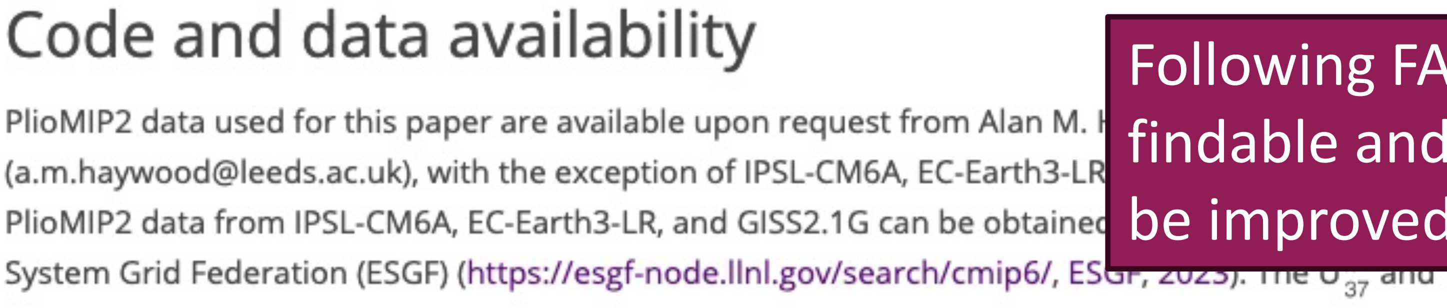
Filter with Facets

General

Identifiers

- Source ID: Select option(s)
- Institution ID: Select option(s)
- Source Type: Select option(s)
- Experiment ID: midPliocene-eo... ×
- Sub Experiment ID: Select option(s)

□	Cart	Dataset ID	Download Options
□	+	CMIP6.PMIP.IPSL.IPSL-CM6A-LR.midPliocene-eoi400.r1i1p1f1.Amon.evspsbl.gr	wget
□	+	CMIP6.PMIP.IPSL.IPSL-CM6A-LR.midPliocene-eoi400.r1i1p1f1.Amon.clwvi.gr	wget
□	+	CMIP6.PMIP.IPSL.IPSL-CM6A-LR.midPliocene-eoi400.r1i1p1f1.Amon.clw.gr	wget
□	+	CMIP6.PMIP.IPSL.IPSL-CM6A-LR.midPliocene-eoi400.r1i1p1f1.Amon.hfss.gr	wget
□	+	CMIP6.PMIP.IPSL.IPSL-CM6A-LR.midPliocene-eoi400.r1i1p1f1.Amon.hfls.gr	wget
□	+	CMIP6.PMIP.IPSL.IPSL-CM6A-LR.midPliocene-eoi400.r1i1p1f1.Amon.clivi.gr	wget

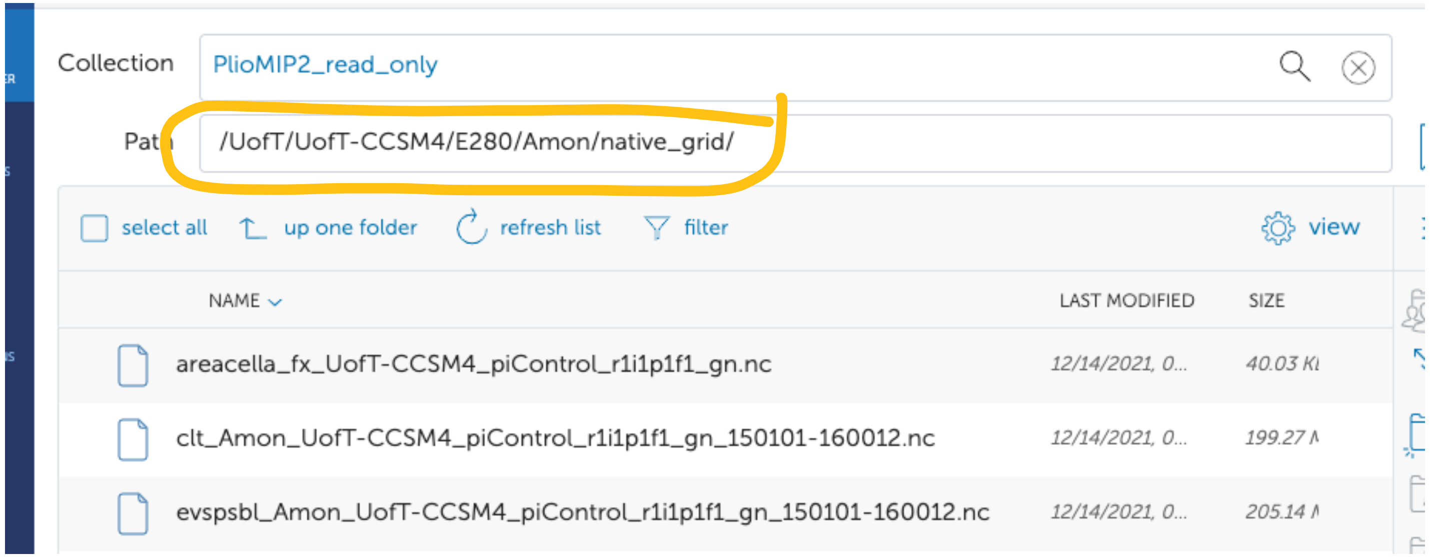


Code and data availability

PlioMIP2 data used for this paper are available upon request from Alan M. H (a.m.haywood@leeds.ac.uk), with the exception of IPSL-CM6A, EC-Earth3-LR PlioMIP2 data from IPSL-CM6A, EC-Earth3-LR, and GISS2.1G can be obtained System Grid Federation (ESGF) (https://esgf-node.llnl.gov/search/cmip6/, ESGF, 2023). The U and

Following FAIR principles… Not very findable and accessible. Could/should be improved!

Step 1: Collecting the data



Collection: PlioMIP2_read_only

Path: /UofT/UofT-CCSM4/E280/Amon/native_grid/

NAME	LAST MODIFIED	SIZE
areacella_fx_UofT-CCSM4_piControl_r1i1p1f1_gn.nc	12/14/2021, 0...	40.03 K
clt_Amon_UofT-CCSM4_piControl_r1i1p1f1_gn_150101-160012.nc	12/14/2021, 0...	199.27 M
evspsbl_Amon_UofT-CCSM4_piControl_r1i1p1f1_gn_150101-160012.nc	12/14/2021, 0...	205.14 M

1b. Download the data

Most models: folder structure
group > model > simulation > atm/ocean > clim/monthly >
1 .nc file for 100 year monthly SSTs.

Nice!

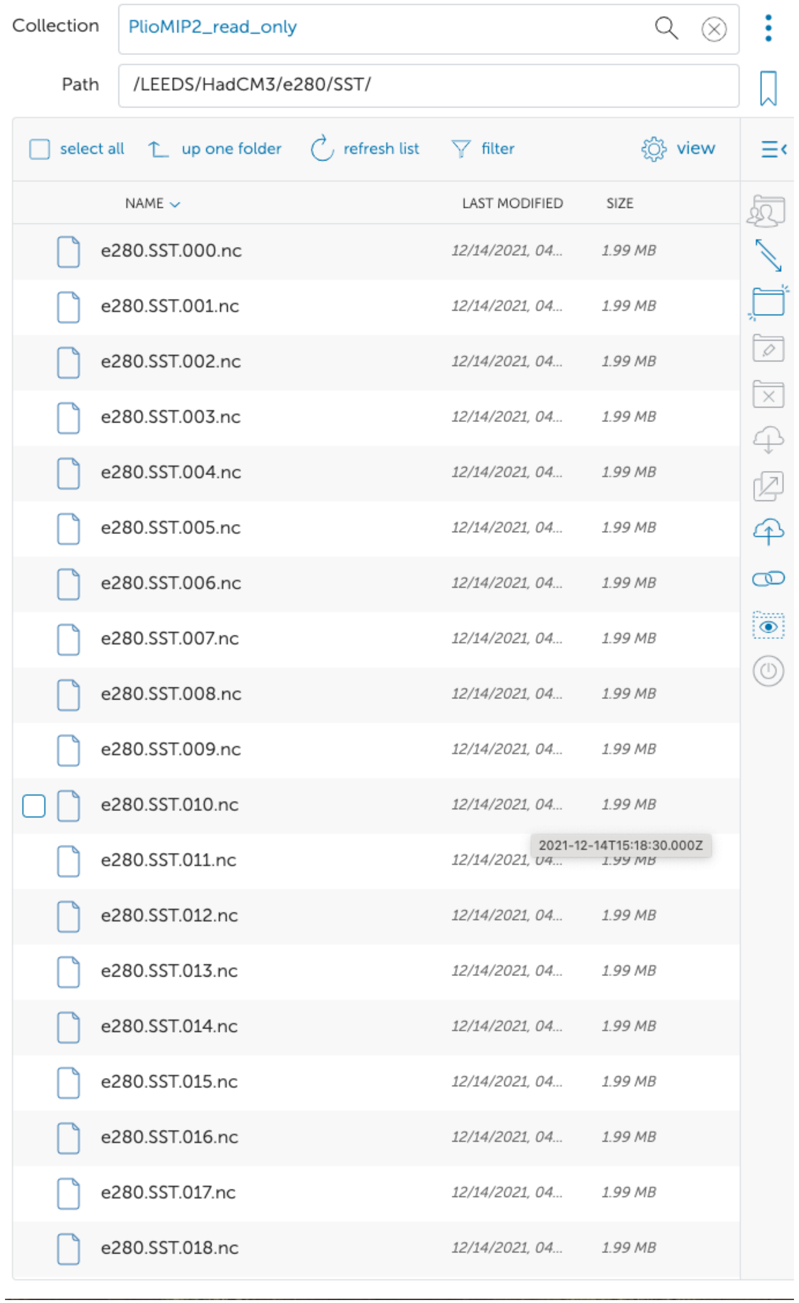
Step 1: Collecting the data

1b. Download the data

Some models… 1 file per year!

No problem if you can run a script that downloads automatically, but…

server only allows for 1 download at the same time! and opens a popup tab every time :'(



Collection: PlioMIP2_read_only

Path: /LEEDS/HadCM3/e280/SST/

NAME	LAST MODIFIED	SIZE
e280.SST.000.nc	12/14/2021, 04...	1.99 MB
e280.SST.001.nc	12/14/2021, 04...	1.99 MB
e280.SST.002.nc	12/14/2021, 04...	1.99 MB
e280.SST.003.nc	12/14/2021, 04...	1.99 MB
e280.SST.004.nc	12/14/2021, 04...	1.99 MB
e280.SST.005.nc	12/14/2021, 04...	1.99 MB
e280.SST.006.nc	12/14/2021, 04...	1.99 MB
e280.SST.007.nc	12/14/2021, 04...	1.99 MB
e280.SST.008.nc	12/14/2021, 04...	1.99 MB
e280.SST.009.nc	12/14/2021, 04...	1.99 MB
□ e280.SST.010.nc	12/14/2021, 04...	1.99 MB
e280.SST.011.nc	12/14/2021, 04...	1.99 MB
e280.SST.012.nc	12/14/2021, 04...	1.99 MB
e280.SST.013.nc	12/14/2021, 04...	1.99 MB
e280.SST.014.nc	12/14/2021, 04...	1.99 MB
e280.SST.015.nc	12/14/2021, 04...	1.99 MB
e280.SST.016.nc	12/14/2021, 04...	1.99 MB
e280.SST.017.nc	12/14/2021, 04...	1.99 MB
e280.SST.018.nc	12/14/2021, 04...	1.99 MB

2021-12-14T15:18:30.000Z

Ok, now we have all the data.

Now we (pre)process it so it is ready for analysis

Step 2: (pre-)processing and standardization

Some (small / annoying) issues:

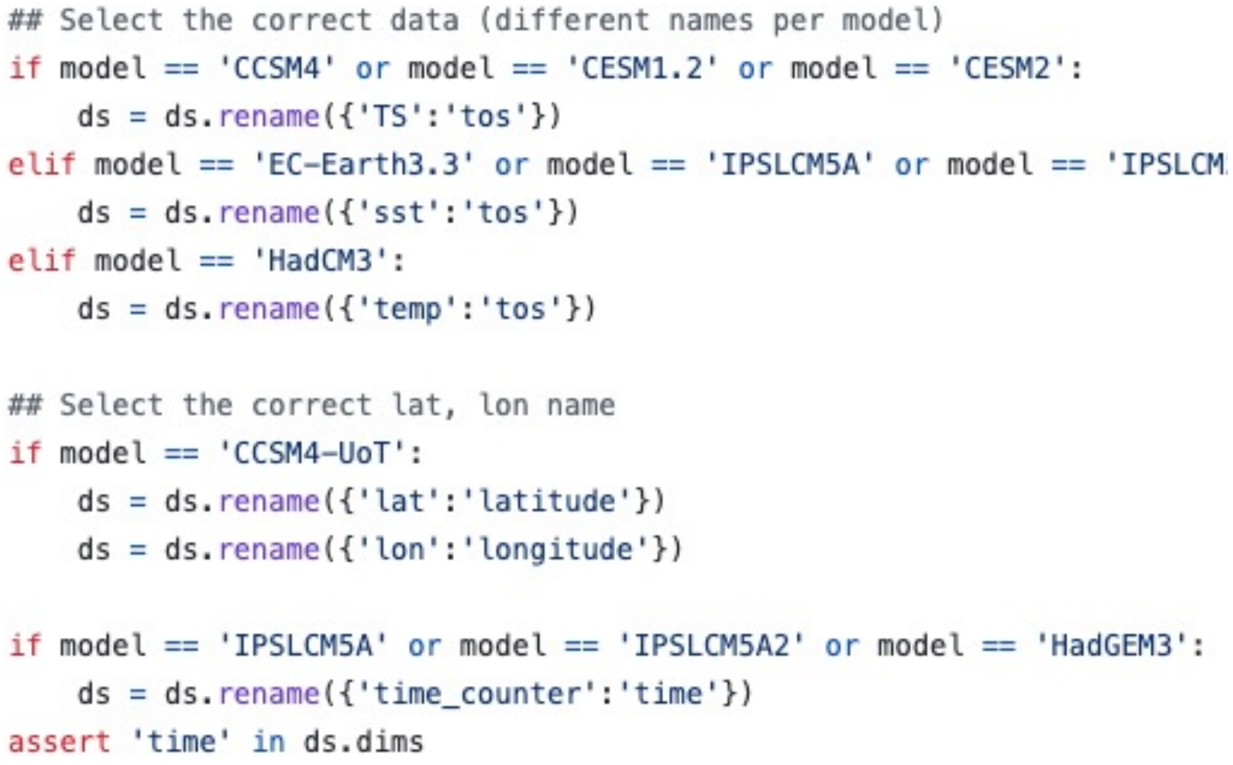
- Very different file names
- Same variables in different units (precipitation in mm/day or m/s or kg/m/s)
- Naming conventions not the same (*lat* or *latitude*, *time* or *month*)
- Time not the same (model years different + monthly mean timed in 15jan or 1feb)
- Lat, lon grid not the same (might require regridding)
- Concat yearly files
- Total precip not always output (convective + large-scale)

Step 2: (pre-)processing and standardization

What I used to do (Do not try this at home):
Loads of if statements

If model A: do ds.lat>xxx
Elif model B: do ds.latitude>xxx

NOT recommended!



```
## Select the correct data (different names per model)
if model == 'CCSM4' or model == 'CESM1.2' or model == 'CESM2':
    ds = ds.rename({'TS':'tos'})
elif model == 'EC-Earth3.3' or model == 'IPSLCM5A' or model == 'IPSLCM
    ds = ds.rename({'sst':'tos'})
elif model == 'HadCM3':
    ds = ds.rename({'temp':'tos'})

## Select the correct lat, lon name
if model == 'CCSM4-UoT':
    ds = ds.rename({'lat':'latitude'})
    ds = ds.rename({'lon':'longitude'})

if model == 'IPSLCM5A' or model == 'IPSLCM5A2' or model == 'HadGEM3':
    ds = ds.rename({'time_counter':'time'})
assert 'time' in ds.dims
```

Step 2: (pre-)processing and standardization

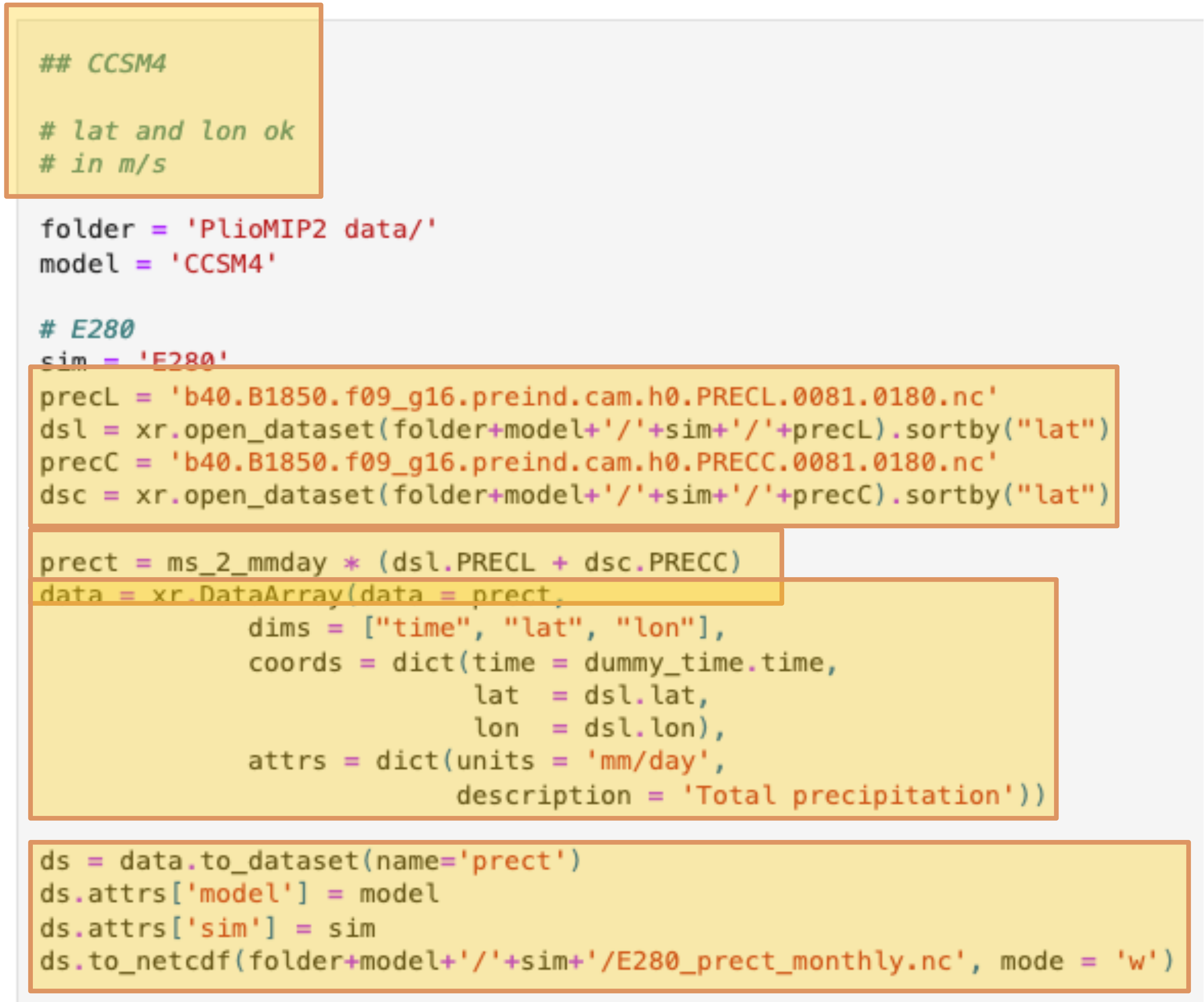
Now: standardize the data BEFORE analysis.

Including:

- Renaming coordinates and variables
- Normalise variable to 1 consistent unit
- Give a ‘dummy’ time
- Resave with same file naming convention

Step 2: (pre-)processing and standardization

Example:



```
## CCSM4

# lat and lon ok
# in m/s

folder = 'PlioMIP2 data/'
model = 'CCSM4'

# E280
sim = 'E280'
precL = 'b40.B1850.f09_g16.preind.cam.h0.PRECL.0081.0180.nc'
dsl = xr.open_dataset(folder+model+'/'+sim+'/'+precL).sortby("lat")
precC = 'b40.B1850.f09_g16.preind.cam.h0.PRECC.0081.0180.nc'
dsc = xr.open_dataset(folder+model+'/'+sim+'/'+precC).sortby("lat")

prect = ms_2_mmday * (dsl.PRECL + dsc.PRECC)
data = xr.DataArray(data = prect,
             dims = ["time", "lat", "lon"],
             coords = dict(time = dummy_time.time,
                           lat  = dsl.lat,
                           lon  = dsl.lon),
             attrs = dict(units = 'mm/day',
                          description = 'Total precipitation'))

ds = data.to_dataset(name='prect')
ds.attrs['model'] = model
ds.attrs['sim'] = sim
ds.to_netcdf(folder+model+'/'+sim+'/E280_prect_monthly.nc', mode = 'w')
```

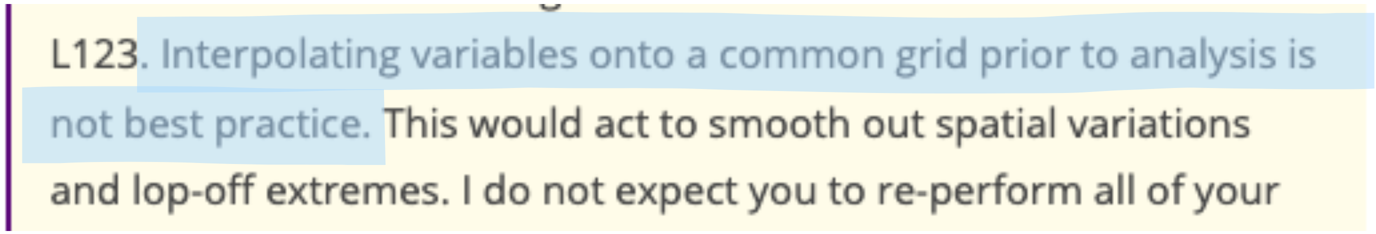
Step 2: (pre-)processing and standardization

Standardization saves me time in analysis! And makes for cleaner codes.

But…

standardizing still costs me and every other researcher using the data time that could have been saved by a clearer protocol (or: groups actually following protocol) for providing/naming the data

NOT Step 2: regridding

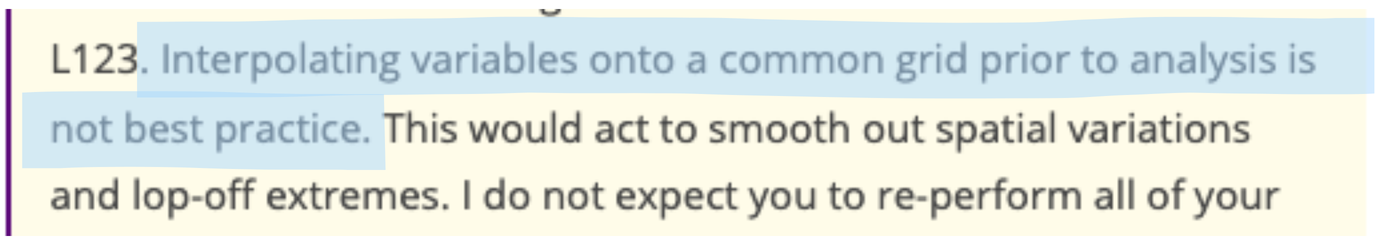


L123. Interpolating variables onto a common grid prior to analysis is not best practice. This would act to smooth out spatial variations and lop-off extremes. I do not expect you to re-perform all of your

What I (learnt to) not do here is: regridding
Why not?

- Regridding should ideally be done at the latest stage possible, i.e. just before you want to compute an ensemble mean to plot it.
- Regridding/interpolation can smooth out spatial variations and local extremes (but also suppress certain unreliable gridbox scale features)
- Ultimately you change your data

NOT Step 2: regridding



L123. Interpolating variables onto a common grid prior to analysis is not best practice. This would act to smooth out spatial variations and lop-off extremes. I do not expect you to re-perform all of your

What I (learnt to) not do here is: regridding
Why not?

- Relevant since PlioMIP2 has varying ocn/atm resolution (roughly ranging from 1x1deg to 3.5x3.5deg!)
- For some analyses, I need an area-averaged time series (El Nino index). Then, regridding is not necessary and can only introduce unnecessary biases.

- Ok.. Must admit: OK for atm (rectangular), ocn more challenging

Ok, now we standardized the data.

Now we can analyze it!

Step 3: Analyze data

Not the main focus of this talk, but will highlight two examples

Ensemble mean spatial field
Area averaged time series

Step 3: Analyze data

Def interpolate ref

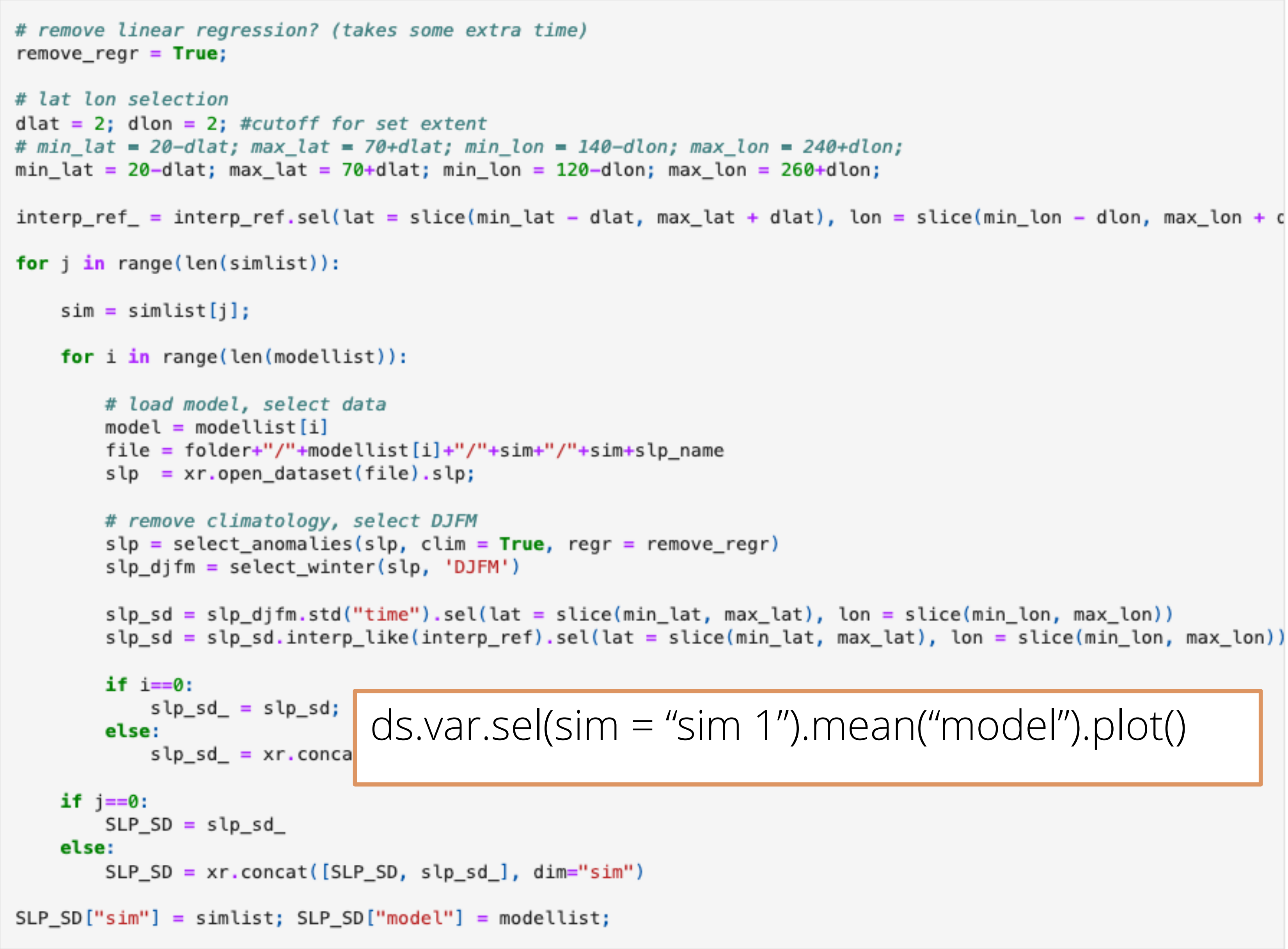
Loop over models and sims

Load data. Easy!

Remove clim, select winter

Compute and interpolate

Combine in one ds

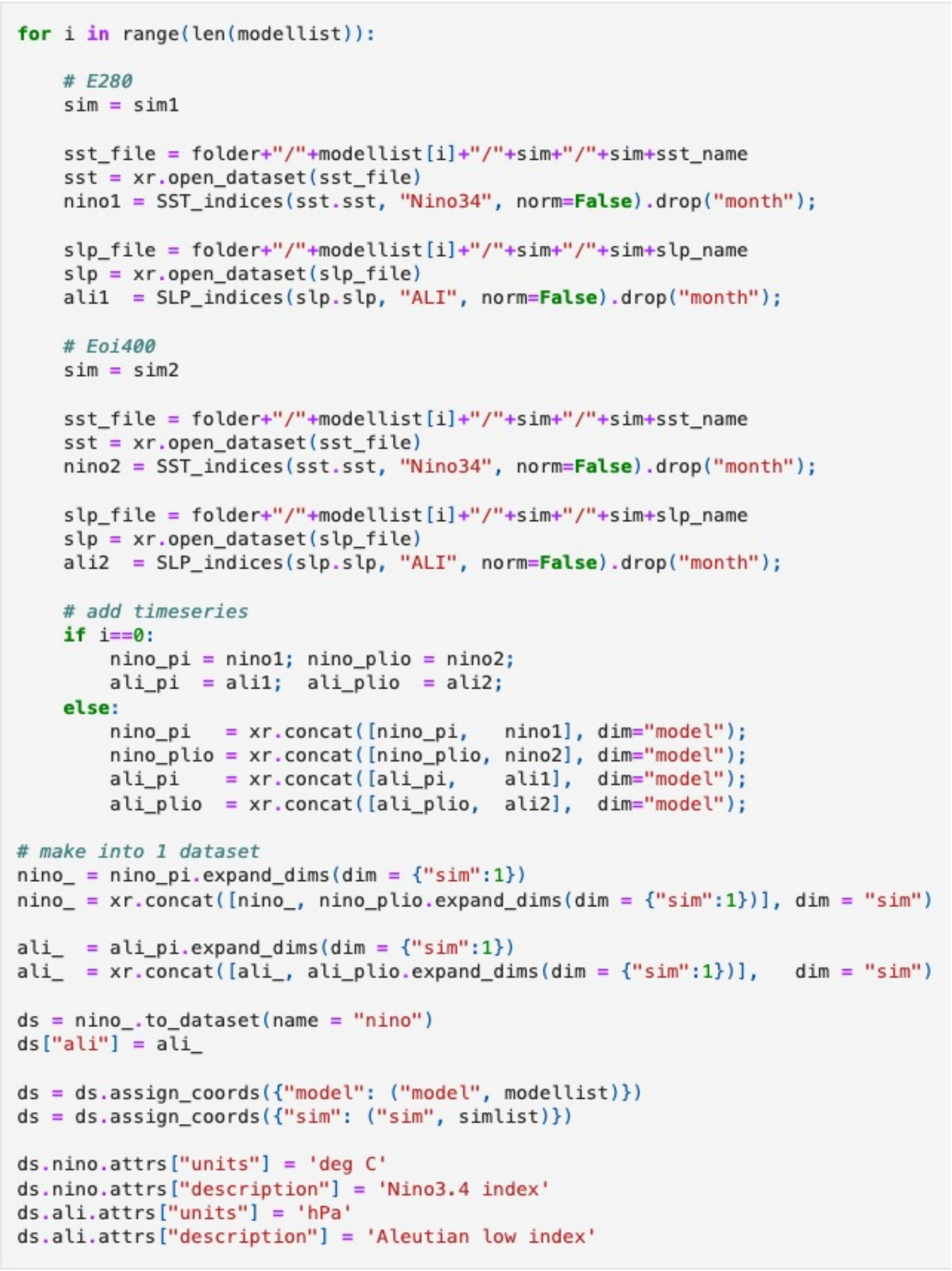


```python
# remove linear regression? (takes some extra time)
remove_regr = True;

# lat lon selection
dlat = 2; dlon = 2; #cutoff for set extent
# min_lat = 20-dlat; max_lat = 70+dlat; min_lon = 140-dlon; max_lon = 240+dlon;
min_lat = 20-dlat; max_lat = 70+dlat; min_lon = 120-dlon; max_lon = 260+dlon;

interp_ref_ = interp_ref.sel(lat = slice(min_lat - dlat, max_lat + dlat), lon = slice(min_lon - dlon, max_lon + c

for j in range(len(simlist)):

    sim = simlist[j];

    for i in range(len(modellist)):

        # load model, select data
        model = modellist[i]
        file = folder+"/"+modellist[i]+"/"+sim+"/"+sim+slp_name
        slp  = xr.open_dataset(file).slp;

        # remove climatology, select DJFM
        slp = select_anomalies(slp, clim = True, regr = remove_regr)
        slp_djfm = select_winter(slp, 'DJFM')

        slp_sd = slp_djfm.std("time").sel(lat = slice(min_lat, max_lat), lon = slice(min_lon, max_lon))
        slp_sd = slp_sd.interp_like(interp_ref).sel(lat = slice(min_lat, max_lat), lon = slice(min_lon, max_lon))

        if i==0:
            slp_sd_ = slp_sd;
        else:
            slp_sd_ = xr.conca

    if j==0:
        SLP_SD = slp_sd_
    else:
        SLP_SD = xr.concat([SLP_SD, slp_sd_], dim="sim")

SLP_SD["sim"] = simlist; SLP_SD["model"] = modellist;
```

ds.var.sel(sim = “sim 1”).mean(“model”).plot()

Step 3: Analyze data

Calculate El Nino indices

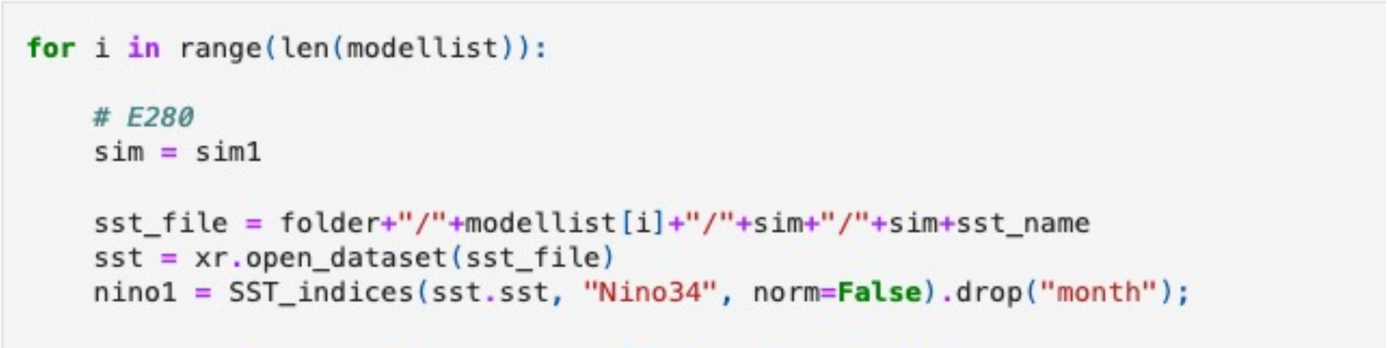


```python
for i in range(len(modellist)):

    # E280
    sim = sim1

    sst_file = folder+"/"+modellist[i]+"/"+sim+"/"+sim+sst_name
    sst = xr.open_dataset(sst_file)
    nino1 = SST_indices(sst.sst, "Nino34", norm=False).drop("month");

    slp_file = folder+"/"+modellist[i]+"/"+sim+"/"+sim+slp_name
    slp = xr.open_dataset(slp_file)
    ali1  = SLP_indices(slp.slp, "ALI", norm=False).drop("month");

    # Eoi400
    sim = sim2

    sst_file = folder+"/"+modellist[i]+"/"+sim+"/"+sim+sst_name
    sst = xr.open_dataset(sst_file)
    nino2 = SST_indices(sst.sst, "Nino34", norm=False).drop("month");

    slp_file = folder+"/"+modellist[i]+"/"+sim+"/"+sim+slp_name
    slp = xr.open_dataset(slp_file)
    ali2  = SLP_indices(slp.slp, "ALI", norm=False).drop("month");

    # add timeseries
    if i==0:
        nino_pi = nino1; nino_plio = nino2;
        ali_pi  = ali1;  ali_plio  = ali2;
    else:
        nino_pi   = xr.concat([nino_pi,   nino1], dim="model");
        nino_plio = xr.concat([nino_plio, nino2], dim="model");
        ali_pi    = xr.concat([ali_pi,    ali1],  dim="model");
        ali_plio  = xr.concat([ali_plio,  ali2],  dim="model");

# make into 1 dataset
nino_ = nino_pi.expand_dims(dim = {"sim":1})
nino_ = xr.concat([nino_, nino_plio.expand_dims(dim = {"sim":1})], dim = "sim")

ali_  = ali_pi.expand_dims(dim = {"sim":1})
ali_  = xr.concat([ali_, ali_plio.expand_dims(dim = {"sim":1})],   dim = "sim")

ds = nino_.to_dataset(name = "nino")
ds["ali"] = ali_

ds = ds.assign_coords({"model": ("model", modellist)})
ds = ds.assign_coords({"sim": ("sim", simlist)})

ds.nino.attrs["units"] = 'deg C'
ds.nino.attrs["description"] = 'Nino3.4 index'
ds.ali.attrs["units"] = 'hPa'
ds.ali.attrs["description"] = 'Aleutian low index'
```

Step 3: Analyze data

Loop over models

Open data (easy) and
compute index per sim



```
for i in range(len(modellist)):

    # E280
    sim = sim1

    sst_file = folder+"/"+modellist[i]+"/"+sim+"/"+sim+sst_name
    sst = xr.open_dataset(sst_file)
    nino1 = SST_indices(sst.sst, "Nino34", norm=False).drop("month");
```

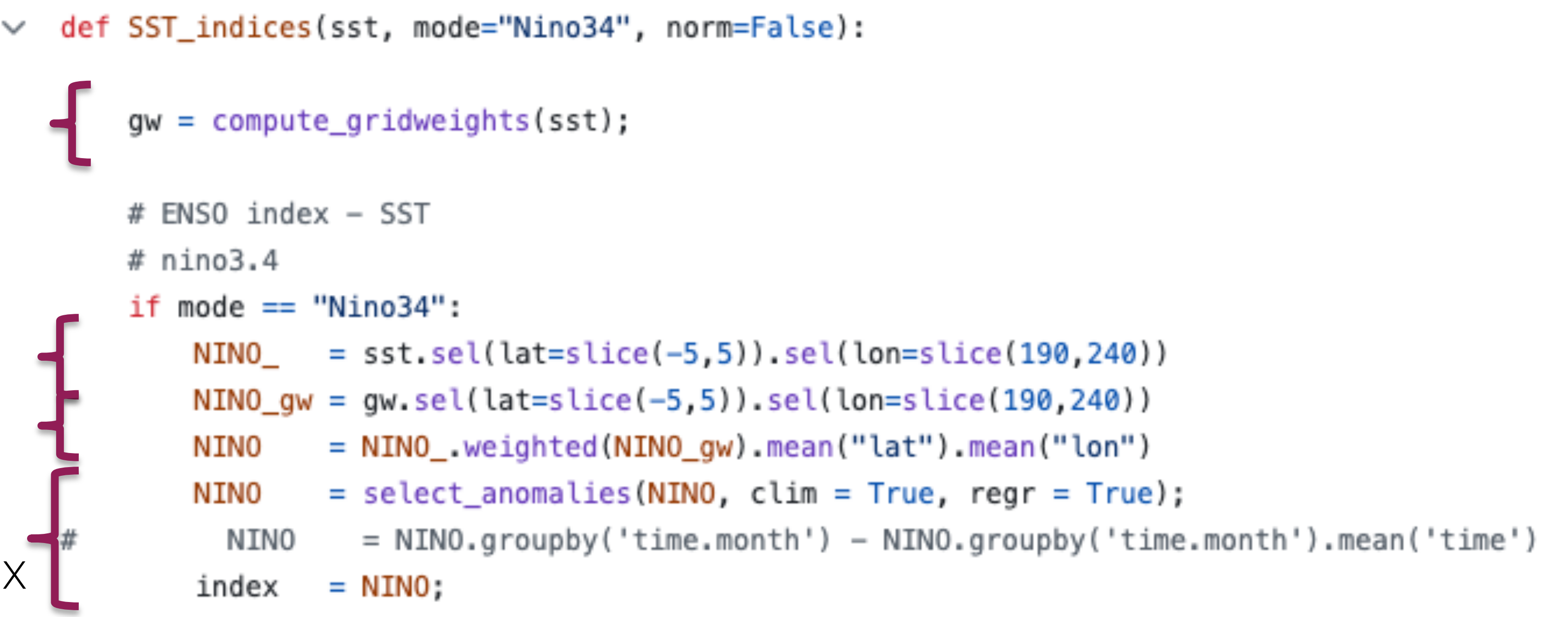
..

Compute gridweights

Select area

Compute weighted
area average

Compute index



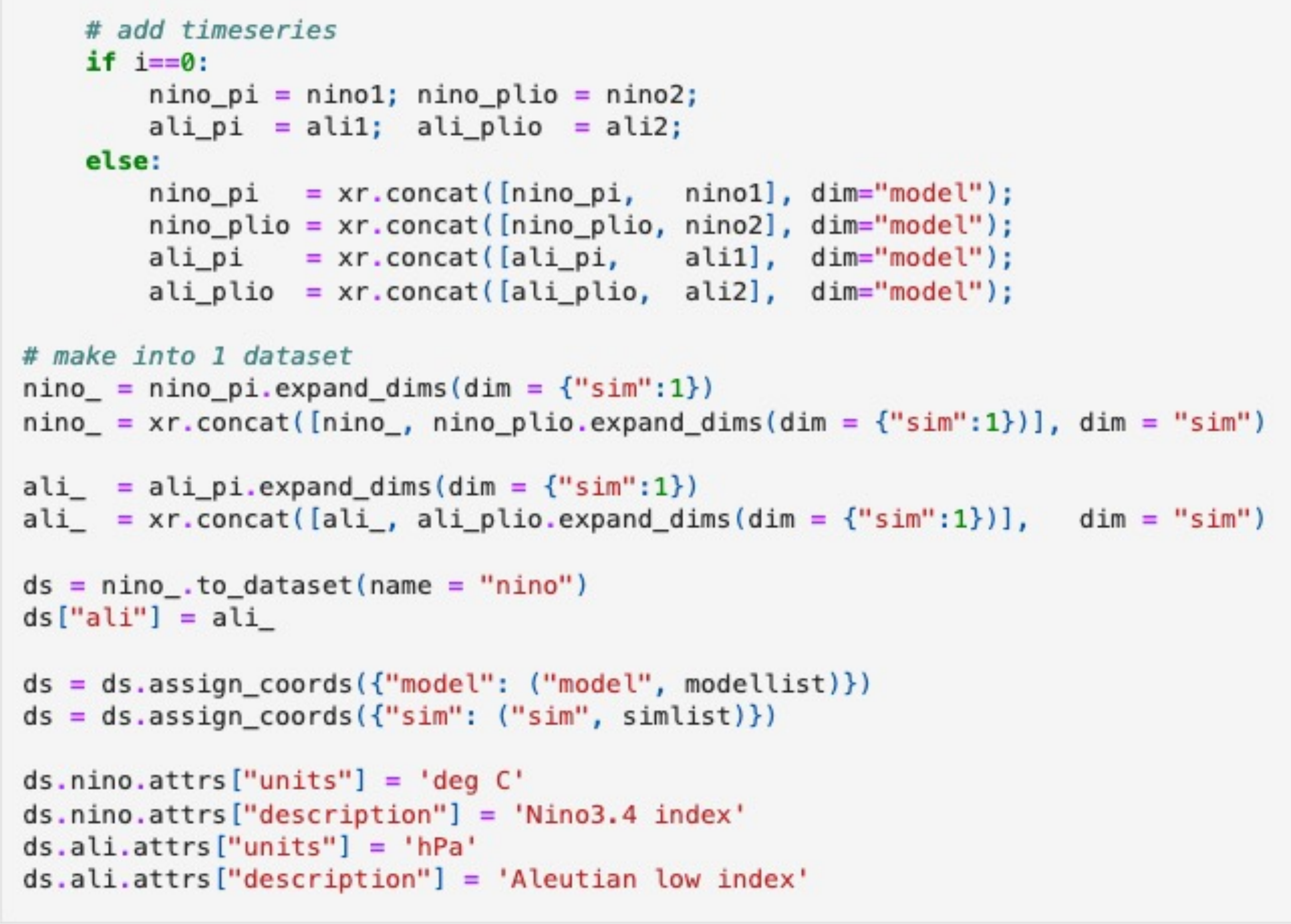
```
def SST_indices(sst, mode="Nino34", norm=False):

    gw = compute_gridweights(sst);

    # ENSO index - SST
    # nino3.4
    if mode == "Nino34":
        NINO_    = sst.sel(lat=slice(-5,5)).sel(lon=slice(190,240))
        NINO_gw = gw.sel(lat=slice(-5,5)).sel(lon=slice(190,240))
        NINO     = NINO_.weighted(NINO_gw).mean("lat").mean("lon")
        NINO     = select_anomalies(NINO, clim = True, regr = True);
#         NINO     = NINO.groupby('time.month') - NINO.groupby('time.month').mean('time')
        index    = NINO;
```

Step 3: Analyze data

..

Concat the data

Making 1 dataset
incl. some info

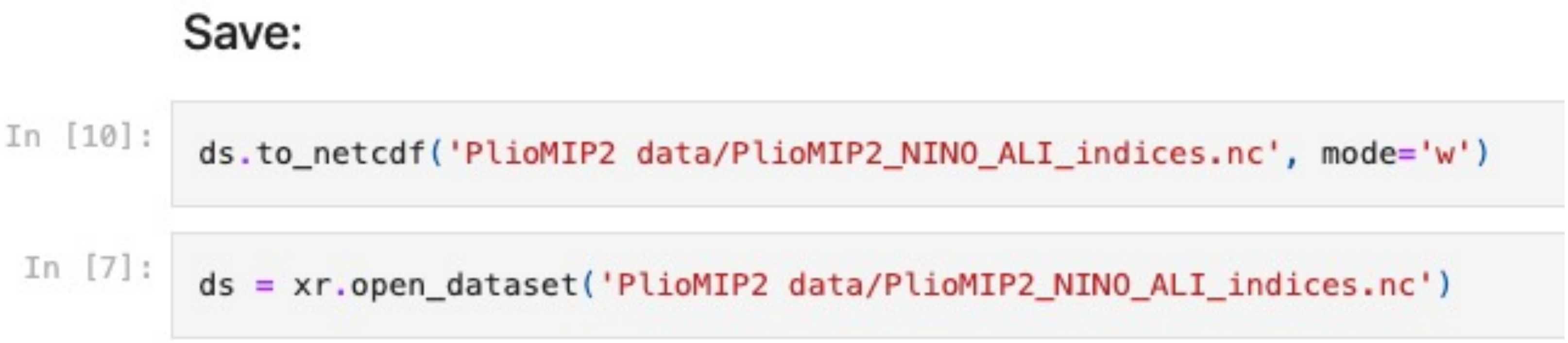


```
    # add timeseries
    if i==0:
        nino_pi = nino1; nino_plio = nino2;
        ali_pi  = ali1;  ali_plio  = ali2;
    else:
        nino_pi   = xr.concat([nino_pi,   nino1], dim="model");
        nino_plio = xr.concat([nino_plio, nino2], dim="model");
        ali_pi    = xr.concat([ali_pi,    ali1],  dim="model");
        ali_plio  = xr.concat([ali_plio,  ali2],  dim="model");

# make into 1 dataset
nino_ = nino_pi.expand_dims(dim = {"sim":1})
nino_ = xr.concat([nino_, nino_plio.expand_dims(dim = {"sim":1})], dim = "sim")

ali_  = ali_pi.expand_dims(dim = {"sim":1})
ali_  = xr.concat([ali_, ali_plio.expand_dims(dim = {"sim":1})],   dim = "sim")

ds = nino_.to_dataset(name = "nino")
ds["ali"] = ali_

ds = ds.assign_coords({"model": ("model", modellist)})
ds = ds.assign_coords({"sim": ("sim", simlist)})

ds.nino.attrs["units"] = 'deg C'
ds.nino.attrs["description"] = 'Nino3.4 index'
ds.ali.attrs["units"] = 'hPa'
ds.ali.attrs["description"] = 'Aleutian low index'
```

Step 3: Analyze data

Save!

& reopen later to save time



Save:

```
In [10]: ds.to_netcdf('PlioMIP2 data/PlioMIP2_NINO_ALI_indices.nc', mode='w')
```

```
In [7]: ds = xr.open_dataset('PlioMIP2 data/PlioMIP2_NINO_ALI_indices.nc')
```

ds.nino.std("time").plot()

Take aways:

- Make sure your data meets FAIR principles
- Design consistent and clear MIP protocol (and try to have model groups follow the conventions!)
- Pls don’t call your January monthly mean “February 01”
- My tip: standardize & resave your ensemble data before analysis
- Regrid at the latest moment (or: don’t)

code from:
https://github.com/arthuroldeman/PlioMIP2-ENSO-teleconnection/tree/main