

ICCLIM & Climate4impact Tool

Christian Pagé
CERFACS, Toulouse, France

Alessandro Spinuso, Ian van der Neut Hans Verhoef, Friedrich Striewski, Mats Veldhuizen
R&D Data Technology and Observations, KNMI

IS-ENES3 Workshop Climate Indices: Eastern European perspective
Monday, May 17 2021



icclim (Index Calculation Climate)

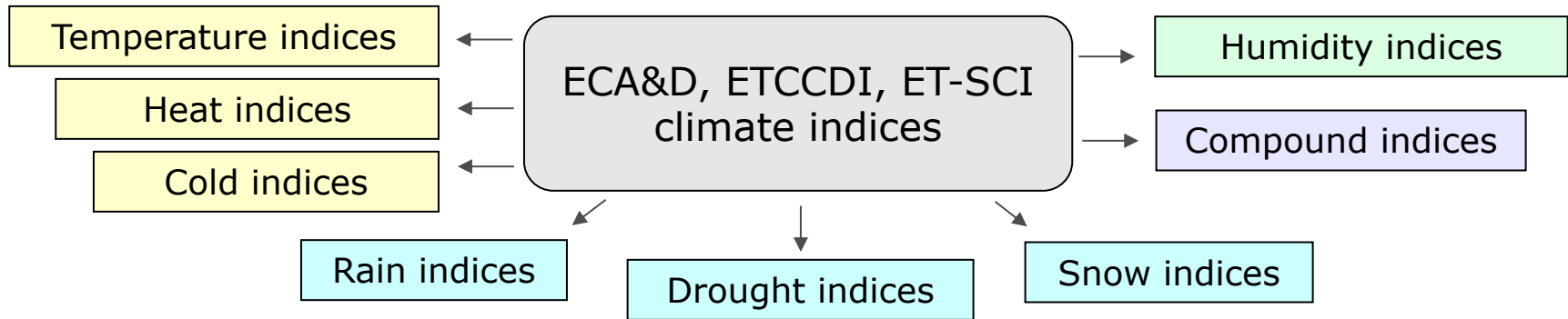
- Developed by CERFACS within FP7 IS-ENES2, H2020 IS-ENES3 and FP7 CLIPC projects
- Written in python, with selected parts optimized in C for very fast performance (version 4.x but newer upcoming 5.x version will instead use xarray/dask)
 - Open Source
 - Support to on-the-fly calculations: fast performance is needed
 - Propagation of Metadata from input to output
 - Adds processing historical information in Metadata (provenance)
 - Comprehensive Documentation
- Can also be used as an independent library in other environments
- ECA&D, ETCCDI climate indices

Documentation: https://icclim.readthedocs.io/en/latest/python_api.html

Source code: <https://github.com/cerfacs-globc/icclim>

Current Version 4.2.18: <https://github.com/cerfacs-globc/icclim/releases/tag/4.2.18>

icclim: climate indices



- Intra-period extreme temperature range [$^{\circ}$ C] - **ETR**
- Warm days (days with mean temperature > 90th percentile of daily mean temperature) - **TG90p**
- Summer days (days with max temperature > 25° C) - **SU**

icclim: climate indices

icclim.indice() - Compute indice

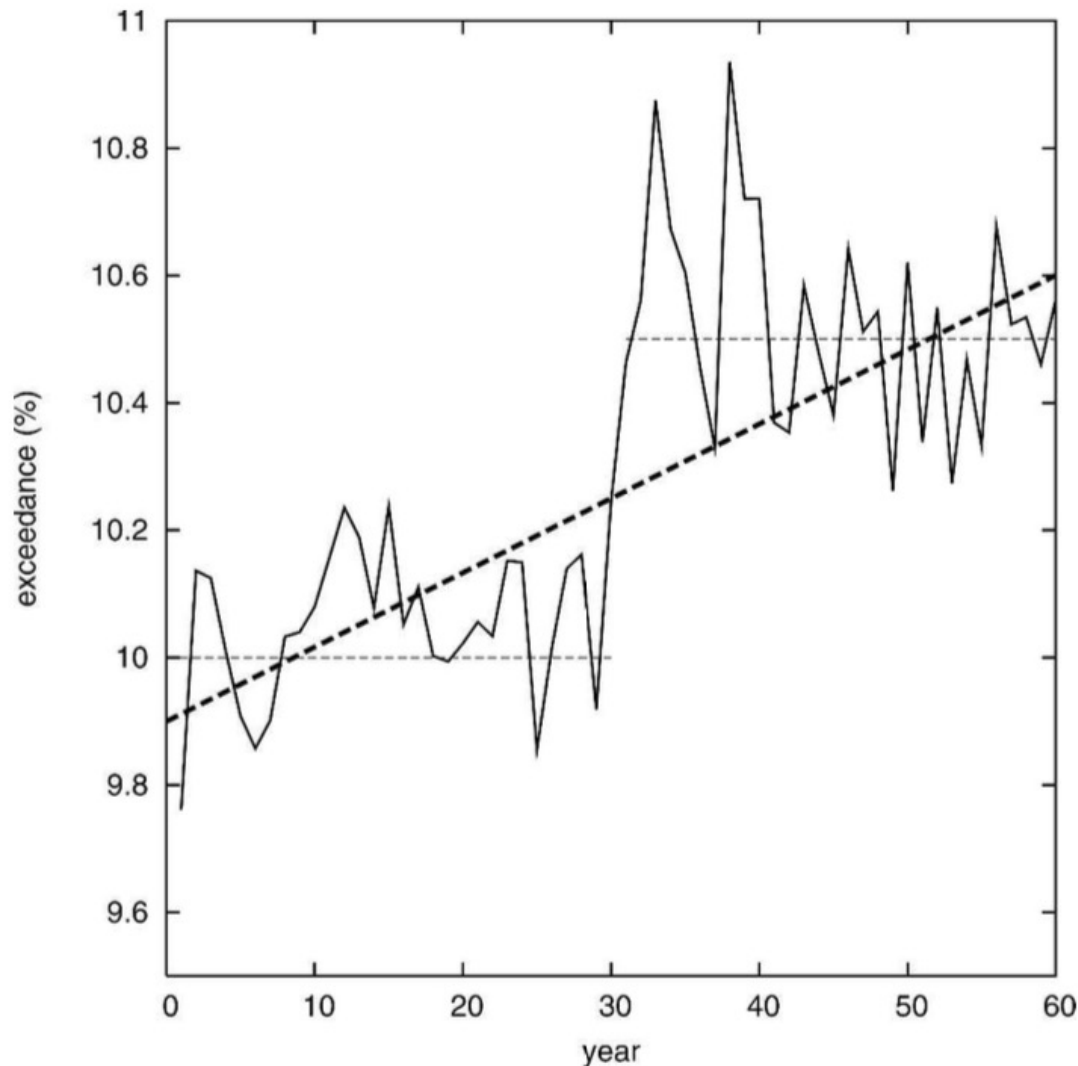
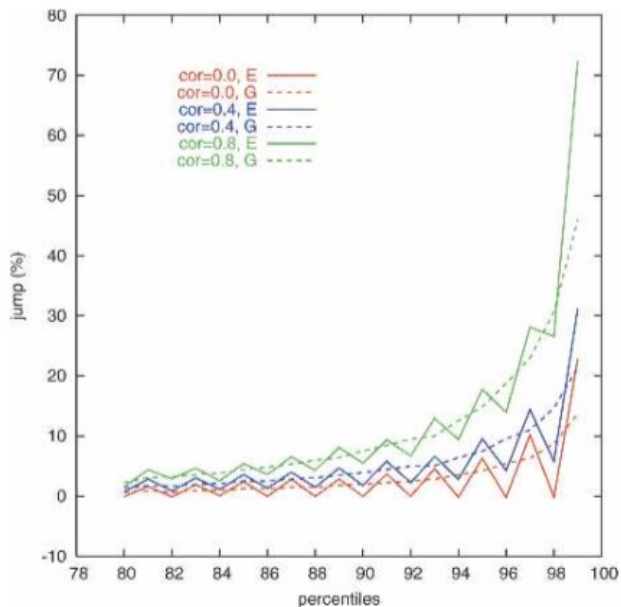
This is the main function to compute an indice:

```
icclim.icclim.indice(in_files, var_name, indice_name=None, slice_mode='year', time_range=None,
out_file='./icclim_out.nc', threshold=None, N_lev=None, lev_dim_pos=1, transfer_limit_Mbytes=None,
callback=None, callback_percentage_start_value=0, callback_percentage_total=100,
base_period_time_range=None, window_width=5, only_leap_years=False, ignore_Feb29th=False,
interpolation='linear', out_unit='days', netcdf_version='NETCDF3_CLASSIC', user_indice=None,
save_percentile=False)
```

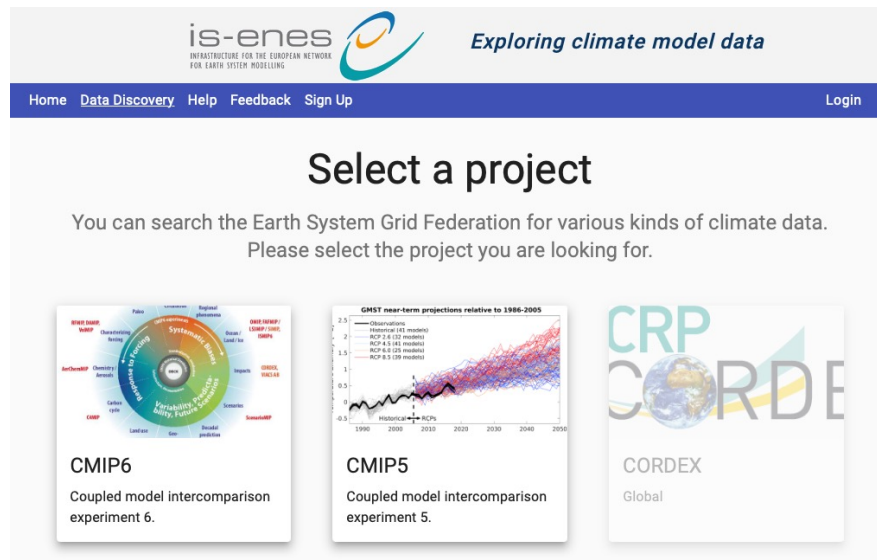
Indice	Source variable
TG, GD4, HD17, TG10p, TG90p	daily mean temperature
TN, TNx, TNn, TR, FD, CFD, TN10p, TN90p, CSDI	daily minimum temperature
TX, TXx, TXn, SU, CSU, ID, TX10p, TX90p, WSDI	daily maximum temperature
DTR, ETR, vDTR	daily maximum + daily minimum temperature
PRCPTOT, RR1, SDII, CWD, CDD, R10mm, R20mm, RX1day, RX5day, R75p, R75pTOT, R95p, R95pTOT, R99p, R99pTOT	daily precipitation flux (liquide phase)
SD, SD1, SD5cm, SD50cm	daily snowfall flux (solid phase)
CD, CW, WD, WW	daily mean temperature + daily precipitation flux (liquide phase)

icclim: climate indices

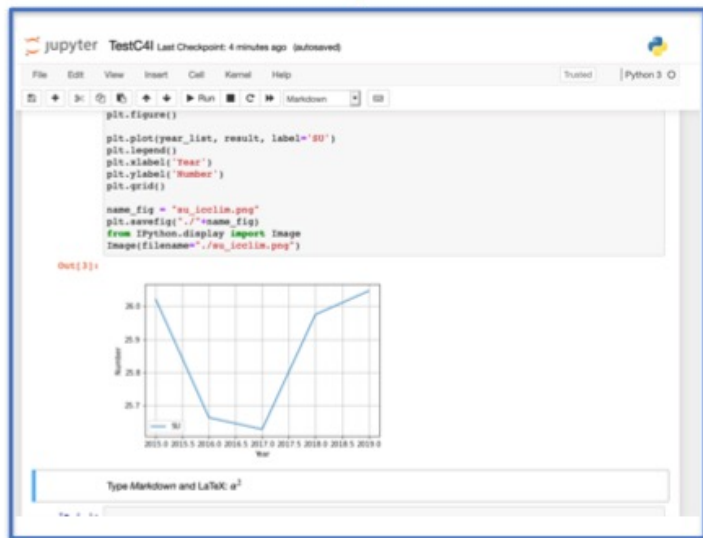
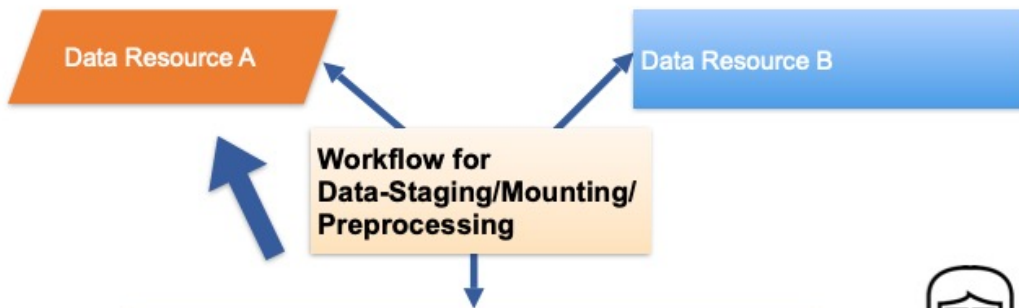
- Implements the bootstrapping method for percentile indices
- Avoid spurious artificial tendencies
- Especially a problem for heavy or extreme values
- Description in Zhang et al. (2005)



- **GUI usability** (Search, Selection, Subsetting)
- **Flexible analysis features** (Integration of Notebooks, ICCLIM and Workflows/Batch-Processing)
- **Automated reproducibility mechanisms** (Datasets Version/GitHub/Binder/Provenance)
- **Making FAIRness of Data and Methods accessible**
- **New front-end technology**
- **Alpha Evaluation Stage:** Will be released soon
- Current Operational C4I 1.0 is available: IS-ENES3 Autumn School Presentations for further information




C4I Workspace Use Case



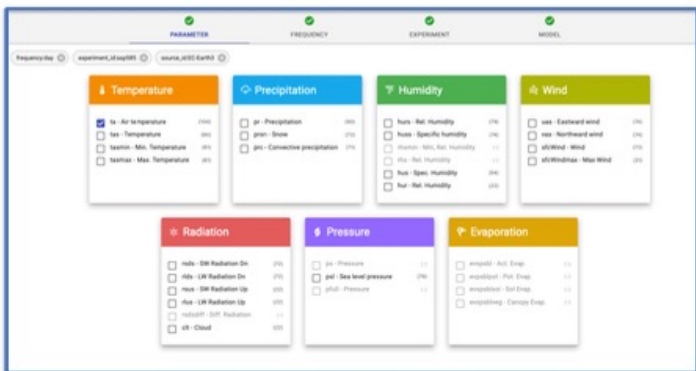
a researcher wants



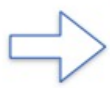
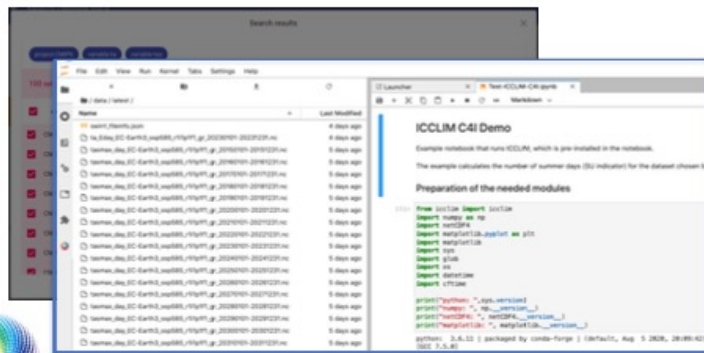
- **access distributed raw data**
- **develop, document and reuse** methods for processing and visualisation.
- **update/extend** raw data and software
- **Track changes and rollback** (Traceability/Recovery)
- **keep old versions of the data** after updates (Reproducibility)
- **snapshot and restore** the state of a workspace software (Reproducibility)

Objective: Extend C4I with Data Driven & Reproducible Workspaces

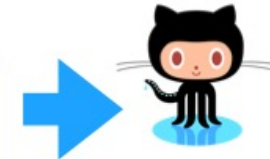
Climate4Impact Search for CMIP5/6
Cordex Data (Distributed Data)



Incremental data staging/subsetting onto customisable
and Reproducible Notebooks (extensible to other tools..)



SWIRRL-API



Software and Environment to Git



MyBinder
Reproduce

Trace Changes to Software and Data
Restore Environments

Data

SWIRRL Jupyter Lab Extension

Monitor
Jobs

Snapshot
Controls

Trace
Activities
and trigger
rollback
actions

The screenshot displays the SWIRRL Jupyter Lab interface. The left sidebar contains several sections: 'Notebook idle', 'Github' (with a 'SWITCH USER' button), 'Snapshot' (with a 'CREATE SNAPSHOT' button and 'Snapshot created: repository url.'), and 'Activities' (with a 'LOAD ACTIVITIES' button). Below these is a table of activities:

Type	Created at	Action
Create	2020-11-18 16:59	RESTORE
Update	2020-11-18 17:08	RESTORE
Snapshot	2020-11-18 17:17	
Snapshot	2020-11-18 17:20	
Snapshot	2020-11-18 17:26	

The main area shows a Jupyter notebook with the following code and output:

```

Test-ICCLIM-C4I.pyb
Terminal 1
Code
2020-11-18 17:33:03,933 .....
2020-11-18 17:33:06,908 Loading data: chunk 1/8 ...
2020-11-18 17:33:52,622 Loading data: chunk 2/8 ...
2020-11-18 17:34:42,806 Loading data: chunk 3/8 ...
2020-11-18 17:35:28,758 Loading data: chunk 4/8 ...
2020-11-18 17:35:39,935 Loading data: chunk 5/8 ...
2020-11-18 17:35:49,512 Loading data: chunk 6/8 ...
2020-11-18 17:36:18,272 Loading data: chunk 7/8 ...
2020-11-18 17:36:32,132 Loading data: chunk 8/8 ...
2020-11-18 17:36:41,195 .....
2020-11-18 17:36:41,196 *
2020-11-18 17:36:41,199 * icclim V4.2.14 *
2020-11-18 17:36:41,201 *
2020-11-18 17:36:41,203 *
2020-11-18 17:36:41,203 * Wed Nov 18 17:36:41 2020 GMT *
2020-11-18 17:36:41,207 *
2020-11-18 17:36:41,207 * END EXECUTION *
2020-11-18 17:36:41,208 *
2020-11-18 17:36:41,209 * CP SECS = 205.487 *
2020-11-18 17:36:41,210 *
2020-11-18 17:36:41,212 .....

Calculate spatial average
[12]: var = np.reshape(var, (var.shape[0], -1))
      result = np.mean(var, axis=1)
      print(result)
[26.02153 25.663391 25.628197 25.975288 26.046402]

Visualise the results
[13]: plt.figure()
      plt.plot(year_list, result, label='SU')
      plt.legend()
      plt.xlabel('Year')
      plt.ylabel('Number')
      plt.grid()
      name_fig = "su_icclim.png"
      plt.savefig("./"+name_fig)
      from IPython.display import Image
      Image(filename="./su_icclim.png")
[13]:
  
```

The plot shows the 'Number' (Y-axis, ranging from 25.7 to 26.0) versus 'Year' (X-axis, ranging from 2015.0 to 2019.0). The data points are approximately: (2015, 26.0), (2016, 25.66), (2017, 25.63), (2018, 25.98), (2019, 26.05). The line is labeled 'SU'.

Example Notebook

File Edit View Run Kernel Tabs Settings Help

Filter files by name

Name	Last Modified
/	
data	6 days ago
lost+found	6 days ago
C4I_Averaged_Temperature_Anomal...	6 days ago
su_icclim_ACCESS-CM2_hist.nc	6 days ago
su_icclim_ACCESS-CM2.nc	6 days ago
su_icclim_BCC-CSM2-MR_hist.nc	6 days ago
su_icclim_BCC-CSM2-MR.nc	6 days ago
su_icclim_CMCC-ESM2_hist.nc	6 days ago
su_icclim_CMCC-ESM2.nc	6 days ago
su_icclim_GFDL-ESM4_hist.nc	6 days ago
su_icclim_GFDL-ESM4.nc	6 days ago
su_icclim_INM-CM5-0_hist.nc	6 days ago
su_icclim_INM-CM5-0.nc	6 days ago
su_icclim_MPI-ESM1-2-LR_hist.nc	6 days ago
su_icclim_MPI-ESM1-2-LR.nc	6 days ago
Welcome to SWIRRL.md	6 days ago

C4I_Averaged_Temperatur... Welcome to SWIRRL.md

Markdown Python 3

ICCLIM C4I: Calculate the Averaged Temperature Anomaly 2081-2100 vs 1971-2000 SSP5

Example notebook that runs ICCLIM, which is pre-installed in the notebook.

The example calculates the averaged temperature anomaly (using the TG indicator) for the period 2081-2100 compared to the reference 1971-2000 for SSP585 and several climate models. It needs the tas parameter chosen by the user on C4I. Alternatively, it could also be used as an example for any other climate indice.

The data is read using xarray and a plot of the time series averaged over Europe is generated, as well as an average spatial map. Several output types examples are shown.

The datasets that are expected for this notebook are tas parameter (needed to calculate the TG indicator) for several climate models, for the historical (1971-2000) and ssp585 (2081-2100) experiments and for one member. Monthly data is much faster to process than using daily data.

In C4I, you can find all of the data needed for models ACCESS-CM2,BCC-CSM2-MR,CMCC-ESM2,GFDL-ESM4,INM-CM5-0,MPI-ESM1-2-LR. Access the CMIP6 project and download them from the esgf-data3.ceda.ac.uk and esgf3.dkrz.de mirrors.

Preparation of the needed modules

```
[33]: import icclim

import sys
import glob
import os
import datetime
import cftime

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import xarray as xr
import nc_time_axis
import rioxarray
import cartopy.crs as ccrs

print("python: ", sys.version)
print("numpy: ", np.__version__)
print("xarray: ", xr.__version__)
```

Simple 0 Python 3 | Idle Mode: Command Ln 1, Col 1 C4I_Averaged_Temperature_Anomaly_2081-2100_vs_1971-2000_SSP5(1).ipynb

Example Notebook

```
[34]: # studied period
dt1 = datetime.datetime(2081,1,1)
dt2 = datetime.datetime(2100,12,16)

# reference period
dtr1 = datetime.datetime(1971,1,1)
dtr2 = datetime.datetime(2000,12,16)

models=['ACCESS-CM2', 'BCC-CSM2-MR', 'CMCC-ESM2', 'GFDL-ESM4', 'INM-CM5-0', 'MPI-ESM1-2-LR']
out_f={}
out_hist_f={}
files=[]
files_hist=[]
for model in models:
    print("Processing model: "+model)
    out_f[model] = 'su_icclim_'+model+'.nc'
    out_hist_f[model] = 'su_icclim_'+model+'_hist.nc'
    files.append('su_icclim_'+model+'.nc')
    files_hist.append('su_icclim_'+model+'_hist.nc')
    #tas_Amon_MPI-ESM1-2-LR_historical_r1i1p1f1_gn_199001-200912.nc
    filenames_hist = glob.glob('./data/latest/tas_Amon_'+model+'_historical_*.nc')
    filenames = glob.glob('./data/latest/tas_Amon_'+model+'_ssp585_*.nc')

icclim.indice(indice_name='TG', in_files=filenames, var_name='tas', slice_mode='year', time_range=[dt1, dt2], transfer_l:
icclim.indice(indice_name='TG', in_files=filenames_hist, var_name='tas', slice_mode='year', time_range=[dtr1, dtr2], tra
```

```
2021-05-10 13:40:43,575 *****
2021-05-10 13:40:43,576 *
2021-05-10 13:40:43,579 *          icclim                                V4.2.17      *
2021-05-10 13:40:43,580 *
2021-05-10 13:40:43,581 *
2021-05-10 13:40:43,581 *      Mon May 10 13:40:43 2021 GMT                    *
2021-05-10 13:40:43,582 *
2021-05-10 13:40:43,582 *      BEGIN EXECUTION                                *
2021-05-10 13:40:43,583 *
2021-05-10 13:40:43,584 *****
```

Processing model: ACCESS-CM2

```
2021-05-10 13:40:43,839 Loading data...
2021-05-10 13:40:44,330 *****
2021-05-10 13:40:44,335 *
2021-05-10 13:40:44,339 *          icclim                                V4.2.17      *
2021-05-10 13:40:44,340 *
2021-05-10 13:40:44,341 *
2021-05-10 13:40:44,341 *      Mon May 10 13:40:44 2021 GMT                    *
2021-05-10 13:40:44,342 *
```

Example Notebook

time360

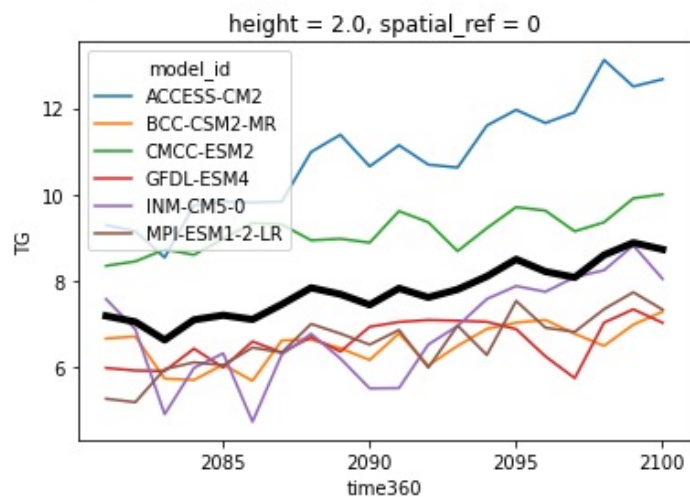
Plot a time series of the anomaly of temperature

Anomaly of temperature for SSP585 of the period 2080-2100 compared to 1971-2000.

The multi-model average is shown in bold black line.

```
[40]: # Plot temperature anomaly compared to historical period and superimpose multi-model average in bold black line
full_tg_anomaly.plot(hue='model_id')
full_tg_anomaly.mean(dim='model_id').plot(color='black', linewidth=4)
```

[40]: [



Example Notebook

ICCLIM C4I: Calculate the number of Summer Days

Example notebook that runs ICCLIM, which is pre-installed in the notebook.

The example calculates the number of summer days (SU indicator) for the dataset chosen by the user on C4I.

The data is read using xarray and a plot of the time series over a specific region is generated, as well as an average spatial map. Several output types examples are shown.

The dataset that is expected for this notebook are tasmax parameter (needed to calculate the SU indicator) for one specific climate model and experiment as well as one member. The time period should be continuous.

To keep this example fast to run, the following period is considered: 2015-01-01 to 2019-12-31, and plots are shown over European region.

Preparation of the needed modules

In [1]:

```
import icclim

import sys
import glob
import os
import datetime
import cftime

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import xarray as xr
import rioxarray
import cartopy.crs as ccrs

print("python: ", sys.version)
print("numpy: ", np.__version__)
print("xarray: ", xr.__version__)
print("pandas: ", pd.__version__)
print("icclim: ", icclim.__version__)
```


Example Notebook

Specification of the parameters and period of interest

In [2]:

```
# studied period
dt1 = datetime.datetime(2015,1,1)
dt2 = datetime.datetime(2019,12,31)

out_f = 'su_icclim.nc'
filenames = glob.glob('./data/latest/tasmax_day*.nc')

icclim.indice(indice_name='SU', in_files=filenames, var_name='tasmax', slice_mode='JJA', time_range=[dt1, dt2], transfer_limit_Mbytes
```

Out [2]:

```
2021-04-07 14:36:25,956 *****
2021-04-07 14:36:25,957 *
2021-04-07 14:36:25,958 *          icclim                      V4.2.17          *
2021-04-07 14:36:25,958 *
2021-04-07 14:36:25,959 *
2021-04-07 14:36:25,959 *          Wed Apr  7 14:36:25 2021 GMT          *
2021-04-07 14:36:25,960 *
2021-04-07 14:36:25,960 *          BEGIN EXECUTION          *
2021-04-07 14:36:25,961 *
2021-04-07 14:36:25,961 *****
2021-04-07 14:36:28,512 Loading data: chunk 1/8 ...
2021-04-07 14:36:35,739 Loading data: chunk 2/8 ...
2021-04-07 14:36:42,434 Loading data: chunk 3/8 ...
2021-04-07 14:36:48,978 Loading data: chunk 4/8 ...
2021-04-07 14:36:55,094 Loading data: chunk 5/8 ...
2021-04-07 14:37:01,426 Loading data: chunk 6/8 ...
2021-04-07 14:37:07,981 Loading data: chunk 7/8 ...
2021-04-07 14:37:14,473 Loading data: chunk 8/8 ...
2021-04-07 14:37:20,686 *****
2021-04-07 14:37:20,686 *
2021-04-07 14:37:20,687 *          icclim                      V4.2.17          *
2021-04-07 14:37:20,688 *
2021-04-07 14:37:20,689 *
2021-04-07 14:37:20,689 *          Wed Apr  7 14:37:20 2021 GMT          *
2021-04-07 14:37:20,690 *
2021-04-07 14:37:20,690 *****
```


Example Notebook

Plot preparation

In [3]:

```
with xr.open_dataset(out_f, decode_times=False) as ds:
    su_xr = ds.rio.write_crs("epsg:4326", inplace=True)
    ds['time'] = xr.decode_cf(ds).time

print(su_xr)

# For later - grab the crs of the data using rioarray
climate_crs = su_xr.rio.crs

# Select a single x,y combination from the data
longitude = su_xr["SU"]["lon"].sel(lon=3.5, method='nearest')
latitude = su_xr["SU"]["lat"].sel(lat=44.2, method='nearest')

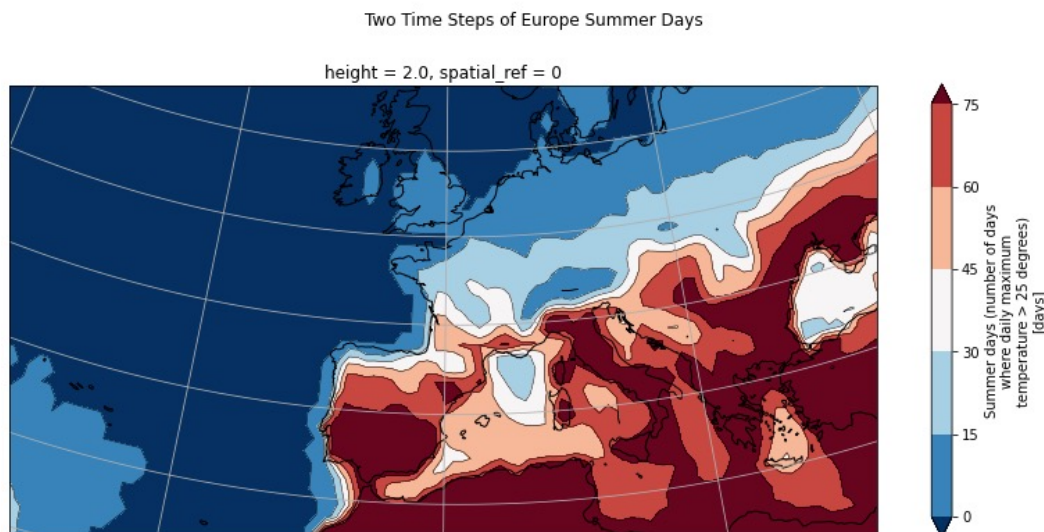
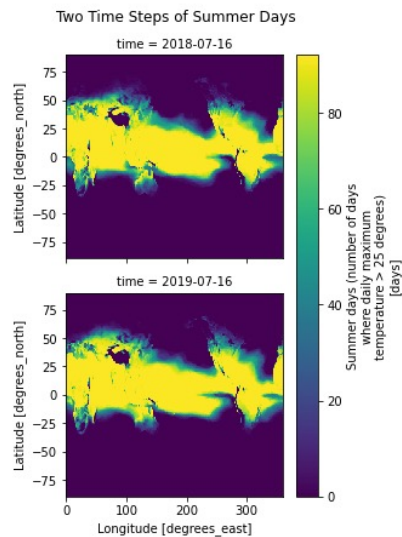
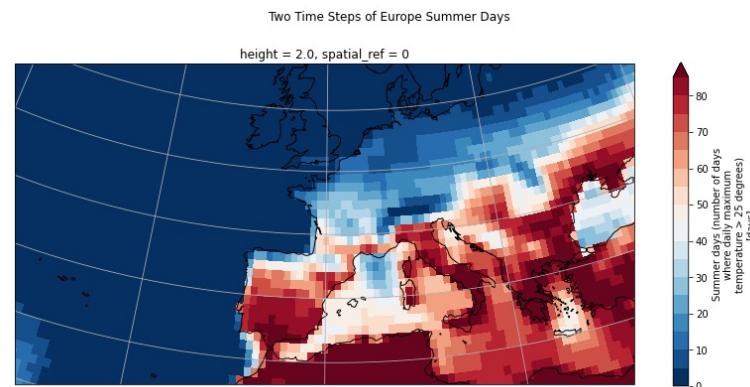
print("Long, Lat values:", longitude, latitude)
```

Out [3]:

```
<xarray.Dataset>
Dimensions:      (bnds: 2, lat: 256, lon: 512, nv: 2, time: 5)
Coordinates:
  * time         (time) datetime64[ns] 2015-07-16 2016-07-16 ... 2019-07-16
  * lat         (lat) float64 -89.46 -88.77 -88.07 -87.37 ... 88.07 88.77 89.46
  * lon         (lon) float64 0.0 0.7031 1.406 2.109 ... 357.9 358.6 359.3
  height       float64 ...
  spatial_ref   int64 0
Dimensions without coordinates: bnds, nv
Data variables:
  time_bnds    (time, bnds) float64 ...
  lat_bnds     (lat, nv) float64 ...
  lon_bnds     (lon, nv) float64 ...
  SU           (time, lat, lon) float32 ...
Attributes:
  title:       ECA heat indice SU
  institution: Climate impact portal (http://climate4impact.eu)
  source:
  references:  ATBD of the ECA indices calculation (http://eca.knmi.nl/do...)
  comment:
```

Example Notebook

	lat	lon	height	spatial_ref	SU
time					
2015-07-16	43.859545	3.515625	2.0	0	66.0
2016-07-16	43.859545	3.515625	2.0	0	70.0
2017-07-16	43.859545	3.515625	2.0	0	67.0
2018-07-16	43.859545	3.515625	2.0	0	79.0
2019-07-16	43.859545	3.515625	2.0	0	48.0





Collection of Notebooks

<https://gitlab.com/is-enes-cdi-c4i/notebooks>

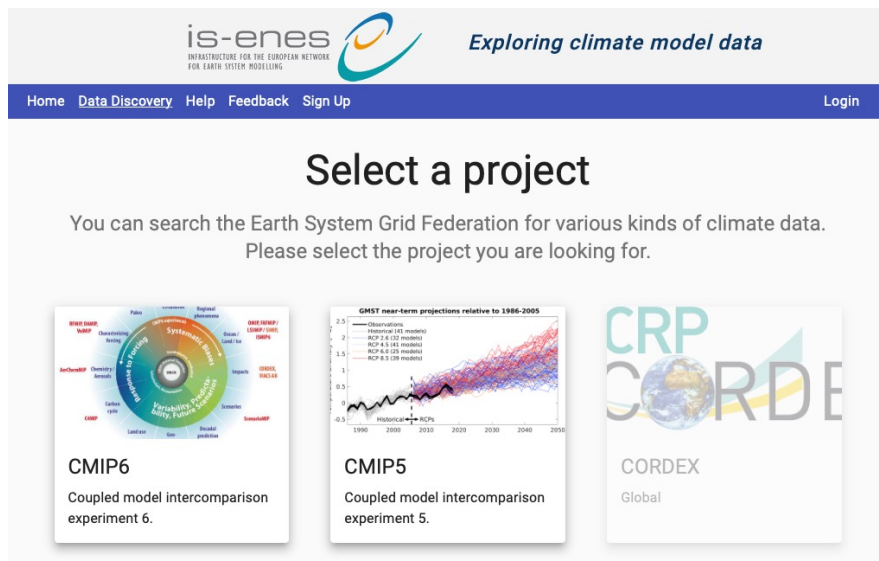
For now 2 Notebooks are available, many more will be available soon

- ICCLIM C4I: Calculate the number of Summer Days
- ICCLIM C4I: Calculate the Averaged Temperature Anomaly 2081-2100 vs 1971-2000 SSP5

Thanks !

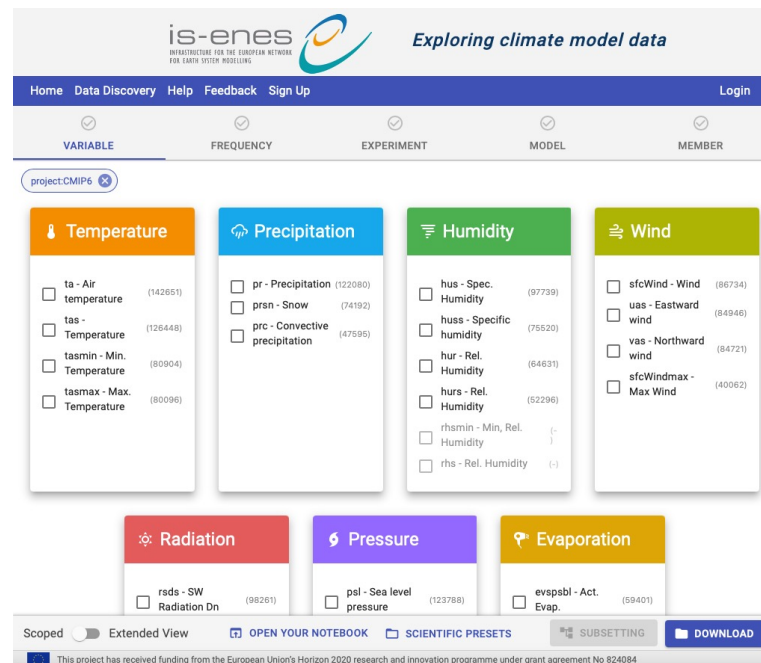
On behalf of the C4I team

- For questions, suggestions, feedback and help, please contact
 - Christian Pagé christian.page@cerfacs.fr
 - Alessandro Spinuso alessandro.spinuso@knmi.nl



The screenshot shows the 'Select a project' page on the is-enes website. It features a navigation bar with 'Home', 'Data Discovery', 'Help', 'Feedback', 'Sign Up', and 'Login'. The main heading is 'Select a project' with the text 'You can search the Earth System Grid Federation for various kinds of climate data. Please select the project you are looking for.' Below this, there are three project cards:

- CMIP6**: Coupled model intercomparison experiment 6. Includes a circular diagram of the Earth System Grid Federation.
- CMIP5**: Coupled model intercomparison experiment 5. Includes a line graph titled 'GMST near-term projections relative to 1986-2005' showing temperature trends from 1996 to 2050.
- CORDEX**: Global. Includes a globe icon.



The screenshot shows the 'Data Discovery' page on the is-enes website. It features a navigation bar with 'Home', 'Data Discovery', 'Help', 'Feedback', 'Sign Up', and 'Login'. Below the navigation bar, there are tabs for 'VARIABLE', 'FREQUENCY', 'EXPERIMENT', 'MODEL', and 'MEMBER'. The 'VARIABLE' tab is selected, showing a search for 'project:CMIP6'. The page displays a grid of variable categories:

- Temperature**: ta - Air temperature (142651), tas - Temperature (126448), tasmin - Min. Temperature (800904), tasmax - Max. Temperature (80096).
- Precipitation**: pr - Precipitation (122080), prsn - Snow (74192), prc - Convective precipitation (47595).
- Humidity**: hus - Spec. Humidity (97739), huss - Specific humidity (75520), hur - Rel. Humidity (64631), hurs - Rel. Humidity (52296), rhamin - Min. Rel. Humidity (-), rhs - Rel. Humidity (-).
- Wind**: sfcWind - Wind (86734), uas - Eastward wind (84946), vas - Northward wind (84721), sfcWindmax - Max Wind (40062).
- Radiation**: rds - SW Radiation Dn (98261).
- Pressure**: psl - Sea level pressure (123788).
- Evaporation**: evspsbl - Act. Evap. (69401).

 At the bottom, there are buttons for 'Scoped', 'Extended View', 'OPEN YOUR NOTEBOOK', 'SCIENTIFIC PRESETS', 'SUBSETTING', and 'DOWNLOAD'. A footer note states: 'This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824084'.