Exercise_2-Solutions

July 19, 2022

`Aim:` *Notebook for the purpose of Climate Risk Assessment:* `Excercise-2 Solutions`

[3]:
```
# import libraries
import cartopy.crs as ccrs # for geographic plotting
import cartopy.feature as cfeature
from IPython.display import Image
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import xclim as xc
import xarray as xr

sns.set(style='ticks')
```

Note: Work with reduced datasets for the exercise tasks, that is the datasets under EUR-11N folder

1 Task 1

To do this task, in your home directory select a file `/lhome/cra2022/YOURHOME/climriskdata/EUR-11N/MPI-M-MP`
`reduced_tas_EUR-11_MPI-M-MPI-ESM-LR_historical_r1i1p1_MPI-CSC-REMO2009_v1_day_19710101-1975123`

Find out the following:

a) Data variables (how many? what are their names?)

b) Coordinates and dimensions of the file. Are coordinates and dimensions the same?

c) Max/Min of the following: time, latitude, longitude

d) Unit of the variable

e) Extract the grid points located close to Bern. (Hint: Use `method='nearest'`)

f) Visualise the time-series for Bern using a quick plot.

[ ]:
```
#adapt path to your own home directory
```

```
input_file = '/lhome/cra2022/o.martius2_2022/climriskdata/EUR-11N/
 ↪MPI-M-MPI-ESM-LR_MPI-CSC-REMO2009_v1/historical/tas/
 ↪reduced_tas_EUR-11_MPI-M-MPI-ESM-LR_historical_r1i1p1_MPI-CSC-REMO2009_v1_day_19710101-1975
 ↪nc'
```

[ ]:
```
ds_tas = xr.open_dataset(input_file)
ds_tas
```

a) `Data variables`: time_bnds, tas

b) `Coordinates`: time, lon, lat

`Dimensions`: Common dimensions for the dataset are bnds, lat, lon, time. Data variable `time_bnds` has time and bnds as dimesions and `tas` has time, lat, lon as dimensions

Dimension has an extra dimension called `bnds`. Apart from that coordinates and dimensions are the same.

c) Max/Min can be checked by using the graphical interface (click on the disk icon)

[ ]:
```
# d)
ds_tas.tas.attrs
```

[ ]:
```
# e and f)
ds_tas.tas.sel(lat=46.9, lon=7.4, method='nearest').plot();
plt.title('Quick time series for Bern');
```

---

2 Task 2

a) Select the years 1971, 1972-1974 from the file you used in task 1.

b) Select the `autumn (SON)` months for the years 1971-1975 (Hint: do it in two steps using a mask).

c) Calculate `mean climatology` for the data selected in b)

d) Calculate anomalies for mean of autumn months in 1973 with respect to climatological mean (1971-1975) and visualize it with a quick plot

[ ]:
```
# a)
ds_tas.sel(time='1971')
```

[ ]:
```
# years 1972- 1974
ds_tas.sel(time=slice('1972', '1974'))
```

[ ]:
```
# b)
# create a time mask
time_mask = ds_tas.time.dt.month.isin([9,10,11]) # returns True or False values
# mask out time
```

```
tas_SON = ds_tas.tas[time_mask,:,:] # since time is first dimension
tas_SON
```

[ ]:
```
# c)
mean_climatology = tas_SON.mean('time')
mean_climatology.plot()
```

[ ]:
```
# Alternate way to do task c is to use groupby function
grouped_data = ds_tas.groupby('time.season')
grouped_data
```

[ ]:
```
grouped_data_mean = grouped_data.mean('time') # will give seasonal mean for all␣
 ↪seasons
grouped_data_mean # a new coordinate and dimension called 'season' is added
```

[ ]:
```
# Now just select the season of interest
# don't forgot to select the variable tas first before plotting
grouped_data_mean.tas.sel(season='SON').plot()
```

[ ]:
```
# d) for anomalies, subtract the mean climatology in task c with tas

# it will be subtracted from all time steps

# note that the dimensions of mean_climatology doesn't have a time dimension
## but xarray will extend it across all dimensions to subtract from each time␣
 ↪step.
### This is based on Numpy's broadcasting principle

# first select the 1973 SON mean
tas_SON_1973 = ds_tas.tas.sel(time=slice('1973-09-01', '1973-11-30')).
 ↪mean('time')

# now subtract with mean climatology ( that is mean for 1971-1975)
anomalies = tas_SON_1973 - mean_climatology
anomalies.plot(cmap='RdBu_r', vmin=-2.0, vmax=+2.0, extend='both')
```

---

3 Task 3

Take a temperature file (e.g. tas.nc).

a) Convert the temperature from `degrees Kelvin` into `degrees Celsius` by subtracting -273.15 from the variable.

b) Correct the `unit` attribute of the modified file to degree Celsius.

c) Select the first time step of the modified file and verify the temperature using a quick plot.

Are the automatic labelling in the colorbar in correct units, that is degree Celsius? Save only the first time step as a `.nc` file to the disk.

d) Convert temperature unit same as in a) for tas.nc but by using `xclim`

e) Check the `unit` attribute in the file. Did `xclim` correct it automatically?

[ ]:
```
# a)
tas_degC = ds_tas.tas - 273.15
print(tas_degC.attrs) # note that attributes of ds_tas.tas are not␣
 ↪automatically copied to the new data array, only the data is
print(ds_tas.tas.attrs)
```

[ ]:
```
# b)
tas_degC.attrs['units'] = 'degC'
print(tas_degC.attrs)
```

[ ]:
```
# c)
sel_data = tas_degC.isel(time=0)
sel_data.plot()
```

[ ]:
```
# delete above variable
del tas_degC
```

[ ]:
```
# d)
tas_degC = xc.units.convert_units_to(ds_tas.tas, 'degC')
print(tas_degC.attrs['units'])
```

So xclim automatically corrects the unit in the metadata as well!

More examples in the documentation of xclim

4 Task 4

Compare the maximum summer temperature of 1972 and 1973 for all the grid points.

Hint: Select the two years first in seperate variables (`Tmax_1972`, `Tmax_1973`) and calculate the difference.

[ ]:
```
# you can break the below into multiple steps but I'm just lazy :)
Tmax_1972 = ds_tas.tas.sel(time='1972').max('time')
Tmax_1973 = ds_tas.tas.sel(time='1973').max('time')

(Tmax_1973 - Tmax_1972).plot() # Max Temperature is higher in most of the areas␣
 ↪in 1973
```

[ ]:
```
del ds_tas
```

5 Task 5

a) Calculate the 16th and 84th quantiles for the `precipitation rate`. *Use a PR file from EUR-11N*

```
[ ]: #adapt path to your own home directory
pr_file = '/lhome/cra2022/o.martius2_2022/climriskdata/EUR-11N/
 ↪MPI-M-MPI-ESM-LR_MPI-CSC-REMO2009_v1/historical/pr/
 ↪reduced_pr_EUR-11_MPI-M-MPI-ESM-LR_historical_r1i1p1_MPI-CSC-REMO2009_v1_day_19710101-19751
 ↪nc'
```

```
[ ]: ds_pr = xr.open_dataset(pr_file)
ds_pr
```

```
[ ]: pr_q16 = ds_pr.pr.quantile(q=0.16, dim='time')
pr_q84 = ds_pr.pr.quantile(q=0.84, dim='time')
```

```
[ ]: pr_q16.plot()
```

```
[ ]: pr_q84.plot()
```

```
[ ]: del pr_file
```

---

6 Advanced: Task 6

Calculate the `number of freezing days` i.e. days with **maximum** temperature below 0°C for all grid points for the file used in Task 3. Watch out for the units!

a) Use `masking concept` to calculate it. The masked file will contain a binary field that is set to `True` if the temperature is below 0°C and to `False` everywhere else. Then use `sum()` to add up the freezing days you have calculated over the whole period and have a quick look at the result with a quick plot.

b) Use `xclim.icclim.ID()` to calculate. Check the results with both degree Celsius and degree Kelvin input file. Does `xclim` takes care of units automatically?

```
[ ]: # using file with daily max data
#adapt path to your own home directory
ds_tas_max = xr.open_dataset('/lhome/cra2022/o.martius_2022/climriskdata/
 ↪EUR-11N/MPI-M-MPI-ESM-LR_MPI-CSC-REMO2009_v1/historical/tasmax/
 ↪reduced_tasmax_EUR-11_MPI-M-MPI-ESM-LR_historical_r1i1p1_MPI-CSC-REMO2009_v1_day_19710101-1
 ↪nc')
```

```
[ ]: tas_max_degC = xc.units.convert_units_to(ds_tas_max.tasmax, 'degC')
tas_max_degC
```

```
[ ]:
```

We used masking in time dimension in Task2 for selecting autumn months. Here we use the same concept for the temperature values.

[ ]:
```
# a)
temp_mask = tas_max_degC < 0 # gives true and false values
count = temp_mask.sum().values # this adds up all True values
print('Total number of days with temperature below 0 deg C are {}'.
 ↪format(count))
```

[ ]:
```
# to have mask days for each grid-point instead, we can visualize it in a quick␣
 ↪2-D map
temp_mask.sum(dim='time').plot()
# note that label on color bar is wrong but xarray just uses the label from␣
 ↪original tas file
```

[ ]:
```
# b) Note that xclim counts number of freezing days for each year by default.
# hence time is now 5
freezing_days = xc.indicators.icclim.ID(tas_max_degC)
freezing_days
```

[ ]:
```
# if we want the same output as xclim with our mask field then there is a small␣
 ↪trick
# we can simply resample our data to YS. YS means "Year Start", that means the␣
 ↪time will be 1st Jan of each year
# resample function always ask how you want to reduce your data, example mean,␣
 ↪max, sum, etc
# in our case taking "sum" makes sense
temp_mask.resample(time='YS').sum()
# you can visually compare the preview output that it is the same as above
```

[ ]:
```
# Now if we now sum the freezing days from xclim's output for all the 5 years␣
 ↪then we will get the same result as above with using masking.
freezing_days.sum('time').plot()
```

[ ]:
```
# does xclim works with tas in deg K?
tas_degK = ds_tas_max.tasmax
freezing_days_K = xc.indicators.icclim.ID(tas_degK)
freezing_days_K.sum(dim='time').plot()
```

[ ]:
```
del ds_tas_max
```

Yes! xclim checks the units from your data's attributes. As long as you keep the units correct in the original file, xclim will automatically handle unit conversion

7 Adanced: Task 7

Compare the values of `Heating Degree Days` and `Tropical Nights Index` for a future temperature scenario with current temperatures. What could that imply for our energy consumption?

[4]:
```
#adapt path to your own home directory
current_file = '/lhome/cra2022/l.quirino.2_2022/climriskdata/EUR-11/
 ↪MPI-M-MPI-ESM-LR_MPI-CSC-REMO2009_v1/historical/tasmax/
 ↪tasmax_EUR-11_MPI-M-MPI-ESM-LR_historical_r1i1p1_MPI-CSC-REMO2009_v1_day_19710101-20001231_
 ↪nc'

future_file = '/lhome/cra2022/l.quirino.2_2022/climriskdata/EUR-11N/
 ↪MPI-M-MPI-ESM-LR_MPI-CSC-REMO2009_v1/rcp85/tasmax/
 ↪reduced_tasmax_EUR-11_MPI-M-MPI-ESM-LR_rcp85_r1i1p1_MPI-CSC-REMO2009_v1_day_20710101-207512
 ↪nc'
```

[5]:
```
ds_tas_current = xr.open_dataset(current_file).sel(time=slice('1996', '2000'),
                                                   lat=slice(44,48),␣
 ↪lon=slice(5,11))
# to reduce file size
ds_tas_future = xr.open_dataset(future_file)
```

Consecutive summer days is counted as the largest number of consecutive days where daily temperature exceeds a threshold. Default threshold is 25 deg C and can be changed by `thresh` argument (see function documentation). Note in some functions the `thresh` argument doesn't work and will give an error. Example if one uses `heating degree days`

[6]:
```
# convert tasmax to deg celsius

current_tas_degC = xc.units.convert_units_to(ds_tas_current.tasmax, 'degC')
future_tas_degC = xc.units.convert_units_to(ds_tas_future.tasmax, 'degC')
```

[7]:
```
# giving custom threshold works with this function
current_CSU = xc.indicators.icclim.CSU(current_tas_degC,)
```

[8]:
```
future_CSU = xc.indicators.icclim.CSU(future_tas_degC,)
```

[ ]:
```
fig = plt.figure(figsize=(8,6))
ax = plt.axes(projection=ccrs.PlateCarree())
(future_CSU.isel(time=0) - current_CSU.isel(time=0)).plot(ax=ax, transform=ccrs.
 ↪PlateCarree(),
                                                          ␣
 ↪cbar_kwargs=dict(label='Difference in CSU',
                                                                ␣
 ↪shrink=0.6)
                                                          )
```

[ ]:
```
del ds_tas_current,ds_tas_future
```

Almost everywhere there is an increase in CSU in the future scenario

7.1 Tropical night index

[ ]:
```
#adapt path to your own home directory
current_file = '/lhome/cra2022/o.martius2_2022//climriskdata/EUR-11/
 ↪MPI-M-MPI-ESM-LR_MPI-CSC-REMO2009_v1/historical/tasmin/
 ↪tasmin_EUR-11_MPI-M-MPI-ESM-LR_historical_r1i1p1_MPI-CSC-REMO2009_v1_day_19710101-20001231_
 ↪nc'

future_file = '/lhome/cra2022/o.martius2_2022//climriskdata/EUR-11N/
 ↪MPI-M-MPI-ESM-LR_MPI-CSC-REMO2009_v1/rcp85/tasmin/
 ↪reduced_tasmin_EUR-11_MPI-M-MPI-ESM-LR_rcp85_r1i1p1_MPI-CSC-REMO2009_v1_day_20710101-207512
 ↪nc'
```

[ ]:
```
ds_tas_current = xr.open_dataset(current_file).sel(time=slice('1996', '2000'),
                                                   lat=slice(44,48),␣
 ↪lon=slice(5,11))
# to reduce file size
ds_tas_future = xr.open_dataset(future_file)
```

[ ]:
```
# convert tasmin to deg celsius

current_tas_degC = xc.units.convert_units_to(ds_tas_current.tasmin, 'degC')
future_tas_degC = xc.units.convert_units_to(ds_tas_future.tasmin, 'degC')
```

[ ]:
```
current_TNI = xc.indicators.icclim.TR(current_tas_degC,)
```

[ ]:
```
future_TNI = xc.indicators.icclim.TR(future_tas_degC,)
```

[ ]:
```
fig = plt.figure(figsize=(8,6))
ax = plt.axes(projection=ccrs.PlateCarree())
current_TNI.isel(time=-1).plot(ax=ax, transform=ccrs.PlateCarree())
ax.add_feature(cfeature.COASTLINE, linestyle=':', color='grey')
ax.add_feature(cfeature.BORDERS, linestyle=':',  color='grey')
#ax.add_feature(cfeature.OCEAN, zorder=10)
```

[ ]:
```
fig = plt.figure(figsize=(8,6))
ax = plt.axes(projection=ccrs.PlateCarree())
future_TNI.isel(time=-1).plot(ax=ax, transform=ccrs.PlateCarree())
ax.add_feature(cfeature.COASTLINE, linestyle=':', color='grey')
ax.add_feature(cfeature.BORDERS, linestyle=':',  color='grey')
#ax.add_feature(cfeature.OCEAN, zorder=10)
```

General Note: Under present day conditions, more than four tropical nights per year occur mainly

in Southern Europe. In Central, northern- and Eastern Europe, tropical nights are very rare events. Under a +2°C global warming, the possible number of tropical nights increases. Projections show that more than 30 tropical nights per year can occur over large areas in Southern Europe. Under present day climate, this happens only in a few restricted areas. Also in Central and Eastern Europe, a couple of tropical nights per year are projected to occur. Areas affected under present day climate in Southern Europe show the largest increase in tropical nights. For a more complete analysis of the impact of tropical nights on health, other factors such as humidity, have to be taken into account.

Source