explore_CMIP6_data

November 4, 2024

1 Explore CMIP6 data on Casper

Ensure you have the required libraries installed, these will make it much easier to work with the data

[3]:
```
#! pip install netcdf4 xarray[io] cartopy nc-time-axis
```

[4]:
```
import pandas as pd
import xarray as xr
import numpy as np
import os.path

import matplotlib.pyplot as plt
# Useful for plotting maps
import cartopy.crs as ccrs

# This can be useful for working with multiple processors - to be explored later on
# from dask.distributed import Client, LocalCluster
```

Output data is in `/glade/collections/cmip/CMIP6/{activity}/NCC/NorESM2-LM/{experiment}`

You can also find other model data here, in particular the NCAR model: Example path: `/glade/collections/cmip/CMIP6/DAMIP/NCAR/CESM2/hist-aer/r1i1p1f1/Amon/tas/gn/latest/*.nc`

Input data is in: `/glade/p/cesmdata/cseg/inputdata/atm/cam/chem/emis/`

The model names are not very obvious but you can either google them, ask ChatGPT, or look them up in these structured dictionaries: https://github.com/PCMDI/cmip6-cmor-tables/tree/main/Tables (which can be queried with e.g. Pandas)

[5]:
```
def get_MIP(experiment):
    """
    Utility function to get the activity associated with a particular experiment
    """
    if experiment == 'ssp245-covid':
        return 'DAMIP'
    elif experiment == 'ssp370-lowNTCF':
        return 'AerChemMIP'
    elif experiment.startswith('ssp'):
        return 'ScenarioMIP'
    elif experiment.startswith('hist-'):
        return 'DAMIP'
    else:
        return 'CMIP'
```

[6]:
```
def get_data(variable, experiment, member):
    """
    Read a particular CMIP6 (Amon) variable from NorESM2
    """
    import glob
    files = glob.glob(f"/glade/collections/cmip/CMIP6/{get_MIP(experiment)}/NCC/
 ↪NorESM2-LM/{experiment}/{member}/Amon/{variable}/gn/v20190815/{variable}/*.
 ↪nc")
    return xr.open_mfdataset(files)[variable]
```

[7]:
```
tas = get_data('tas', 'historical', 'r1i1p1f1')
```

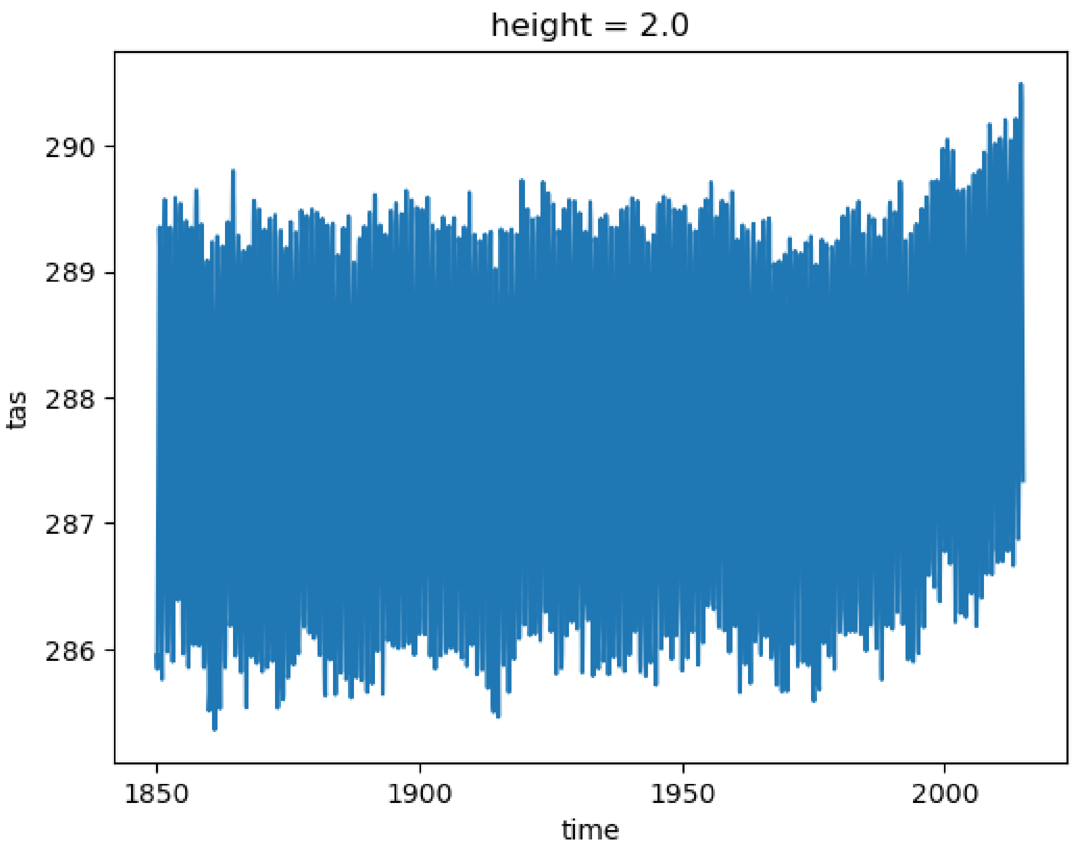
Note, the ensemble member format: `r` for realization, `i` for initialization, `p` for physics, and `f` for forcing

We're only interested in different realizations in this project, so try different r numbers but keep the rest the same: E.g.: `r1i1p1f1`, `r2i1p1f1`, `r3i1p1f1`

[8]:
```
# When averaging gridded data on a sphere, we need to account for the fact that␣
 ↪the values near the poles have less area
weights = np.cos(np.deg2rad(tas.lat))
weights.name = "weights"

tas_timeseries = tas.weighted(weights).mean(['lat', 'lon'])
```

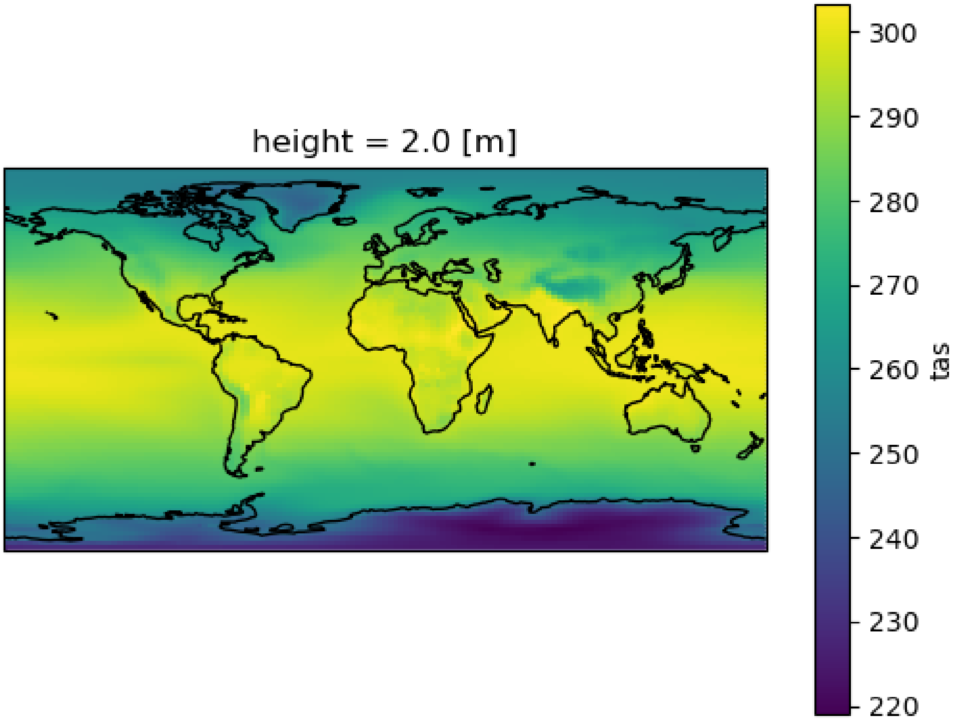
[9]:
```
tas_timeseries.plot()
```

[9]:
```
[<matplotlib.lines.Line2D at 0x14729341dcc0>]
```



[10]:
```
# Plot a map of the average temperature between 1850-1900

tas.sel(time=slice('1850','1900')).mean('time').plot(
    transform=ccrs.PlateCarree(), # This is the projection the data is stored as
    subplot_kws={"projection": ccrs.PlateCarree()}, # This describes the␣
 ↪projection to plot onto (which happens to be the projection the data is␣
 ↪already in so no transformation is needed in this case)
)

# Feel free to explore other projections here: https://scitools.org.uk/cartopy/
 ↪docs/v0.15/crs/projections.html

plt.gca().coastlines()
```

[10]:
```
<cartopy.mpl.feature_artist.FeatureArtist at 0x1472d1e94400>
```

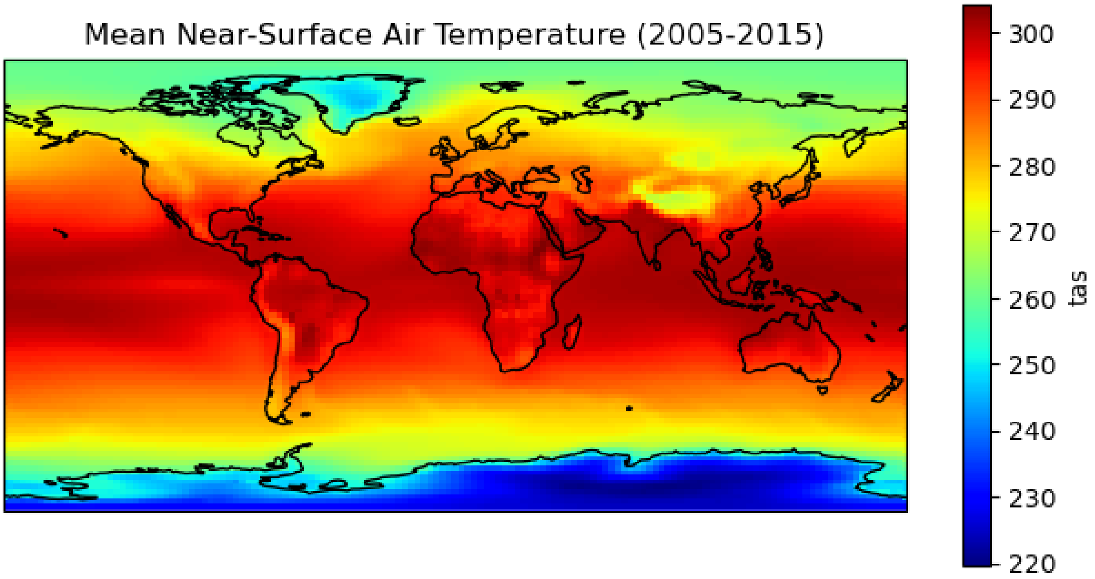


1.0.1 Task to be Completed

Map of the Average Global Temperature between 2005-2015

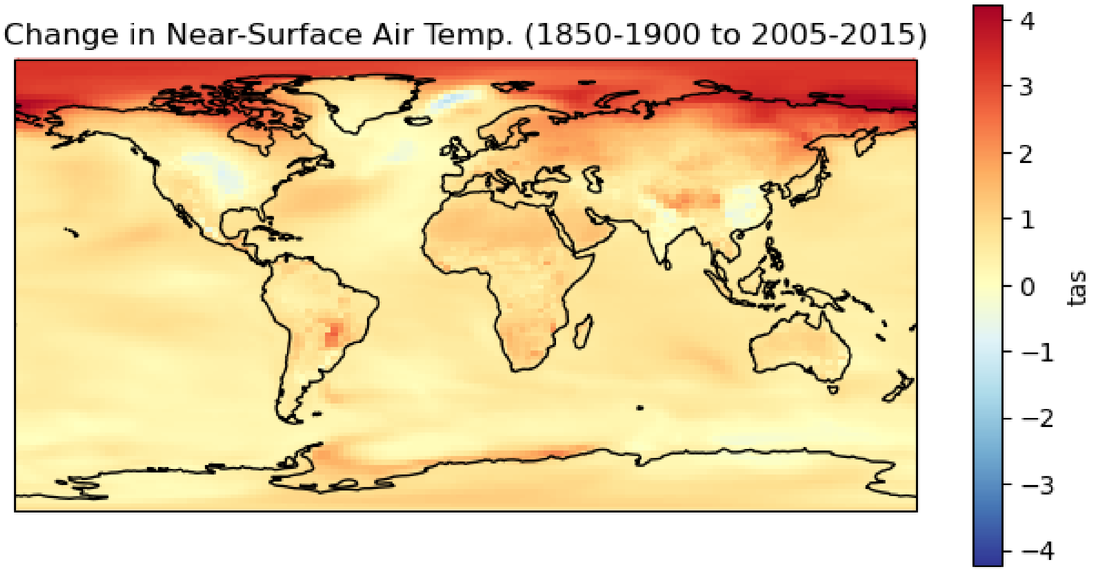
[11]:

```
plt.rcParams['figure.figsize'] = [8, 4]

tas.sel(time=slice('2005','2015')).mean('time').plot(
    transform=ccrs.PlateCarree(),
    subplot_kws={"projection": ccrs.PlateCarree()},
    cmap='jet'
)

plt.title('Mean Near-Surface Air Temperature (2005-2015)')
plt.gca().coastlines()

plt.show()
```



Difference in Global Temperature from 1850-1900 to 2005-2015

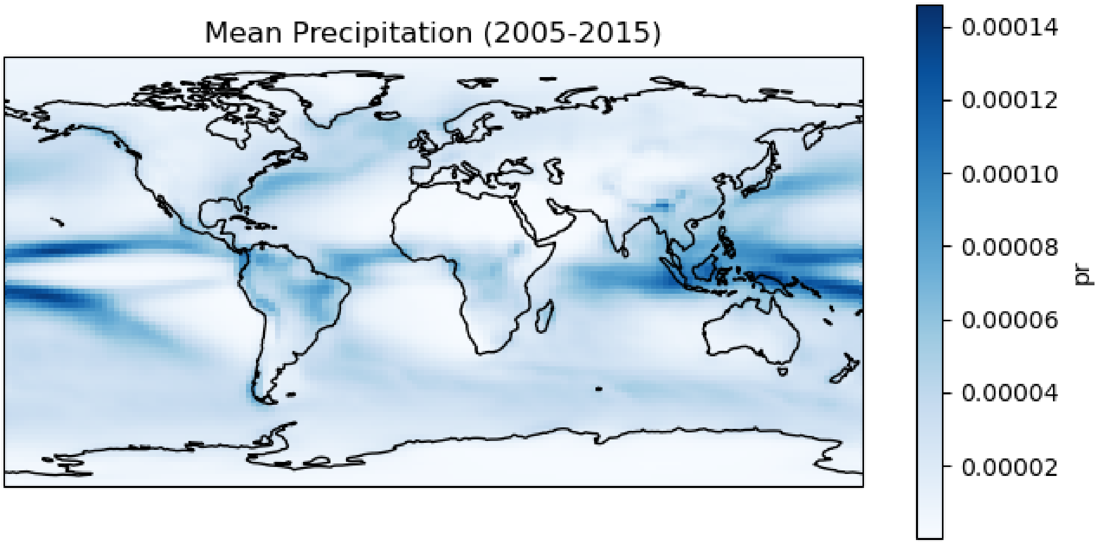
[12]:
```
(tas.sel(time=slice('2005','2015')).mean('time') - tas.
 ↪sel(time=slice('1850','1900')).mean('time')).plot(
    transform=ccrs.PlateCarree(),
    subplot_kws={"projection": ccrs.PlateCarree()},
    cmap="RdYlBu_r",
)

plt.title('Change in Near-Surface Air Temp. (1850-1900 to 2005-2015)')
plt.gca().coastlines()

plt.show()
```



Map of the Average Global Precipitation between 2005-2015

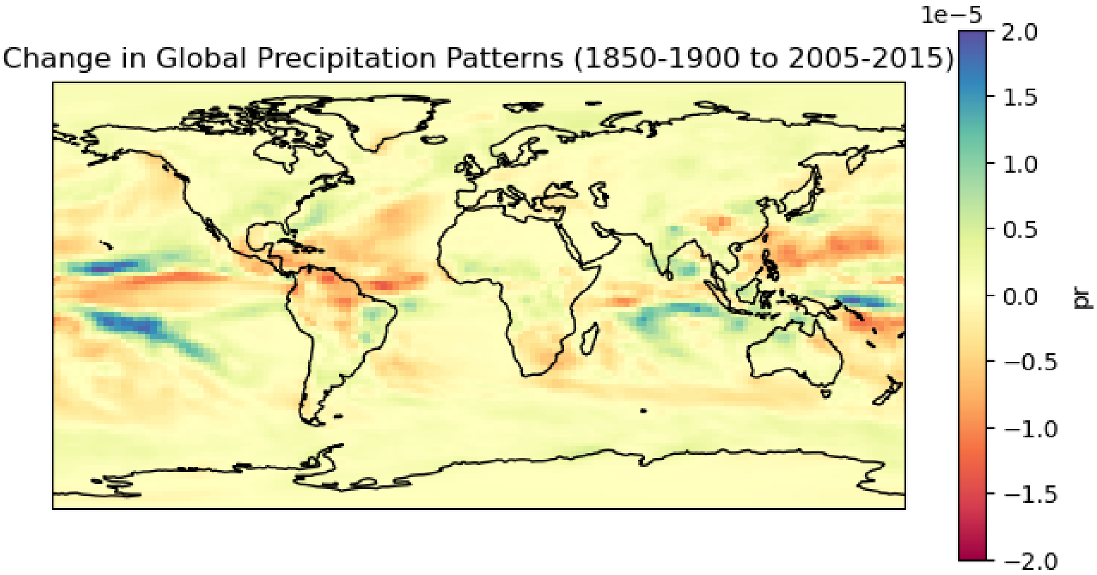
[13]:
```
pr = get_data('pr', 'historical', 'r1i1p1f1')
```

[14]:
```
pr.sel(time=slice('2005','2015')).mean('time').plot(
    transform=ccrs.PlateCarree(),
    subplot_kws={"projection": ccrs.PlateCarree()},
    cmap='Blues'
)

plt.title('Mean Precipitation (2005-2015)')
plt.gca().coastlines()

plt.show()
```



Difference in Global Precipitation from 1850-1900 to 2005-2015

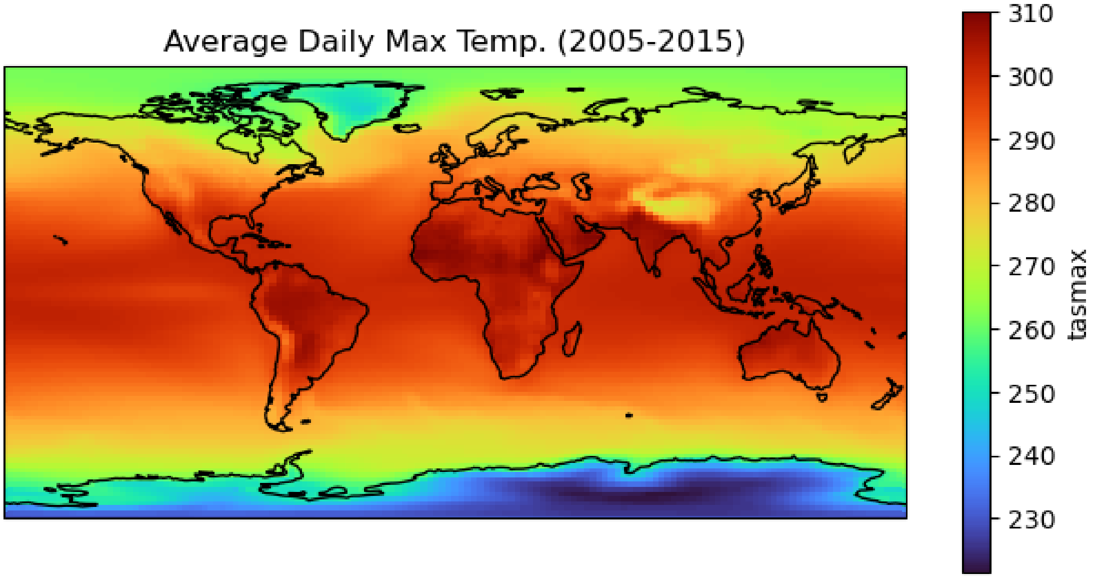
```
[15]: (pr.sel(time=slice('2005','2015')).mean('time') - pr.
       ↪sel(time=slice('1850','1900')).mean('time')).plot(
          transform=ccrs.PlateCarree(),
          subplot_kws={"projection": ccrs.PlateCarree()},
          cmap="Spectral",
      )

      plt.title('Change in Global Precipitation Patterns (1850-1900 to 2005-2015)')
      plt.gca().coastlines()

      plt.show()
```


Change in Global Precipitation Patterns (1850-1900 to 2005-2015)
1e−5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
pr

Map of the Average Daily Max Temperature between 2005-2015

[16]:
```
def get_data_day(variable, experiment, member):
    """
    Read a particular CMIP6 (day) variable from NorESM2
    """
    import glob
    files = glob.glob(f"/glade/collections/cmip/CMIP6/{get_MIP(experiment)}/NCC/
 ↪NorESM2-LM/{experiment}/{member}/day/{variable}/gn/v20190815/{variable}/*.
 ↪nc")
    return xr.open_mfdataset(files)[variable]

tasmax = get_data_day('tasmax', 'historical', 'r1i1p1f1')
```

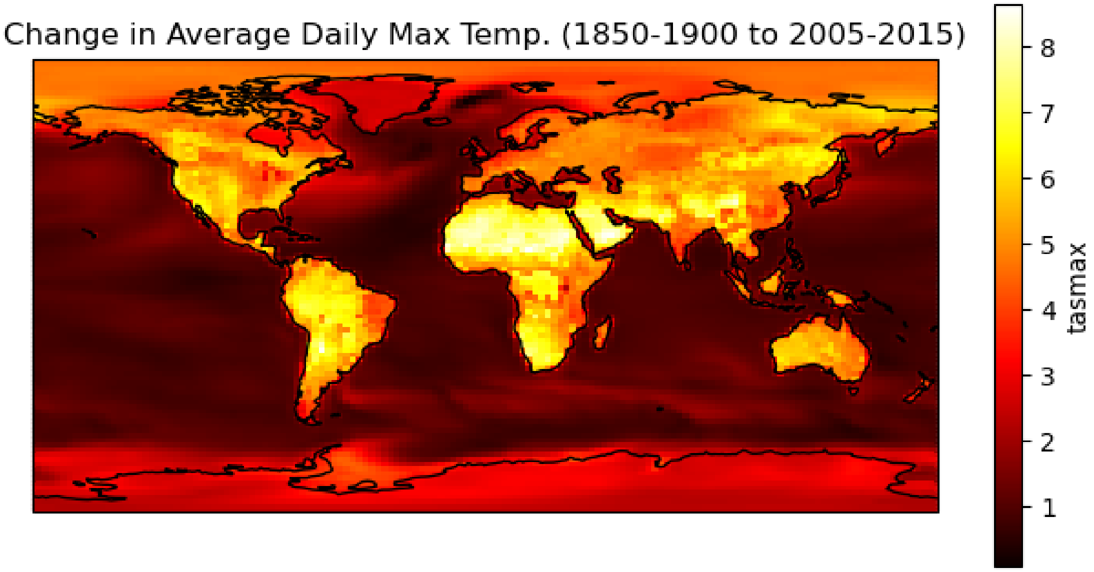
[17]:
```
tasmax.sel(time=slice('2005','2015')).mean('time').plot(
    transform=ccrs.PlateCarree(),
    subplot_kws={"projection": ccrs.PlateCarree()},
    cmap='turbo'
)

plt.title('Average Daily Max Temp. (2005-2015)')
plt.gca().coastlines()

plt.show()
```



Difference in Global Daily Max Temperature from 1850-1900 to 2005-2015

[18]:
```
diff_tasmax = tasmax.sel(time=slice('2005','2015')).mean('time') - tasmax.
↪sel(time=slice('1850','1900')).mean('time')
```

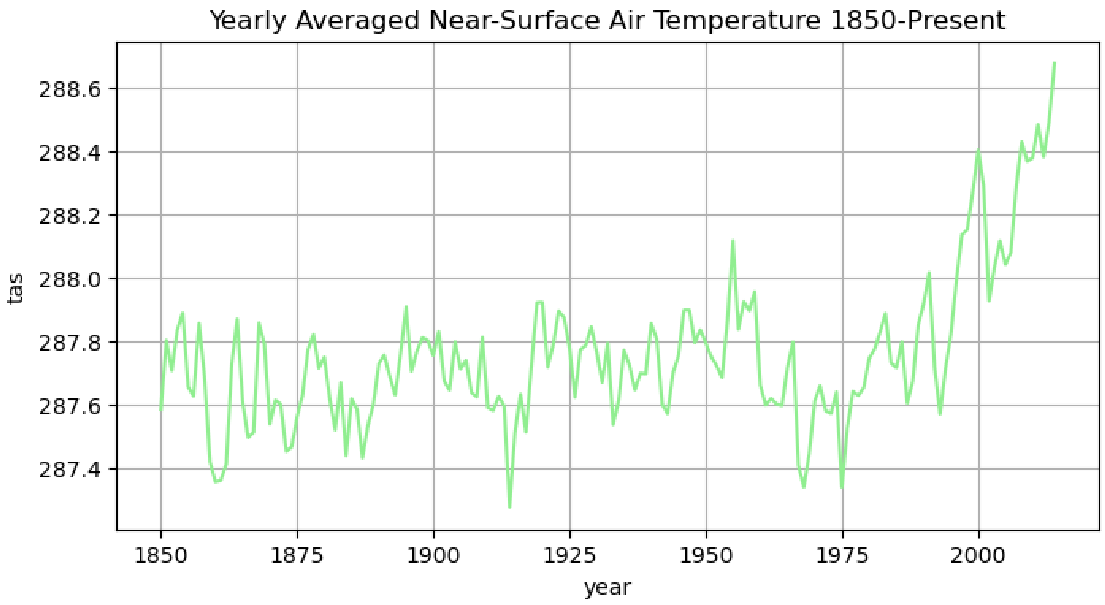
[19]:
```
diff_tasmax.plot(
    transform=ccrs.PlateCarree(),
    subplot_kws={"projection": ccrs.PlateCarree()},
    cmap="hot",
)

plt.title('Change in Average Daily Max Temp. (1850-1900 to 2005-2015)')
plt.gca().coastlines()

plt.show()
```



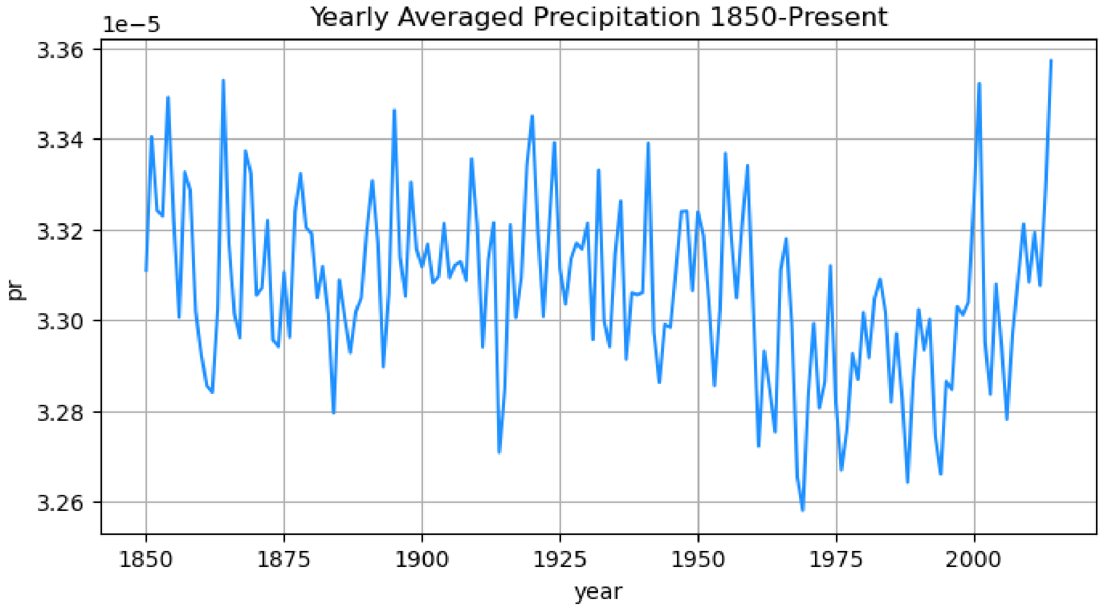
Global Average Time-Series

[20]:
```
weights = np.cos(np.deg2rad(tas.lat))
weights.name = "weights"
```

[21]:
```
tas_timeseries = tas.weighted(weights).mean(['lat', 'lon'])
yearly_tas = tas_timeseries.groupby("time.year").mean()

yearly_tas.plot(color='lightgreen')

plt.grid()
plt.title('Yearly Averaged Near-Surface Air Temperature 1850-Present')

plt.show()
```

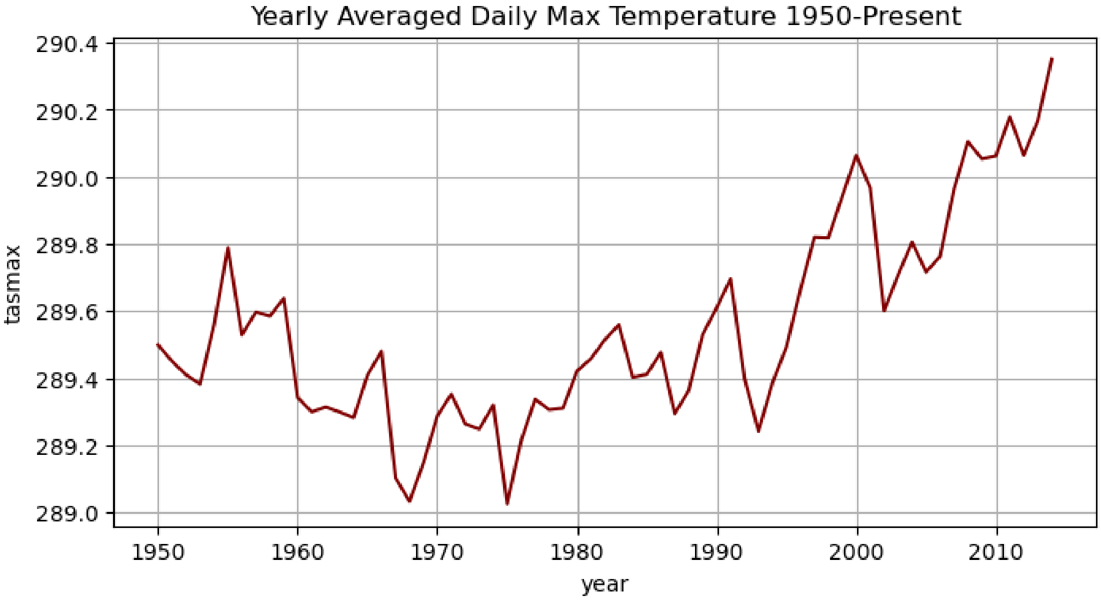


```
[22]: pr_timeseries = pr.weighted(weights).mean(['lat', 'lon'])
      yearly_pr = pr_timeseries.groupby("time.year").mean()

      yearly_pr.plot(color='dodgerblue')

      plt.grid()
      plt.title('Yearly Averaged Precipitation 1850-Present')

      plt.show()
```



[28]:

```
tasmax_timeseries = tasmax.sel(time=slice('1950','2015')).weighted(weights).
  ↪mean(['lat', 'lon'])
yearly_tasmax = tasmax_timeseries.groupby("time.year").mean()

yearly_tasmax.plot(color='maroon')

plt.grid()
plt.title('Yearly Averaged Daily Max Temperature 1950-Present')

plt.show()
```



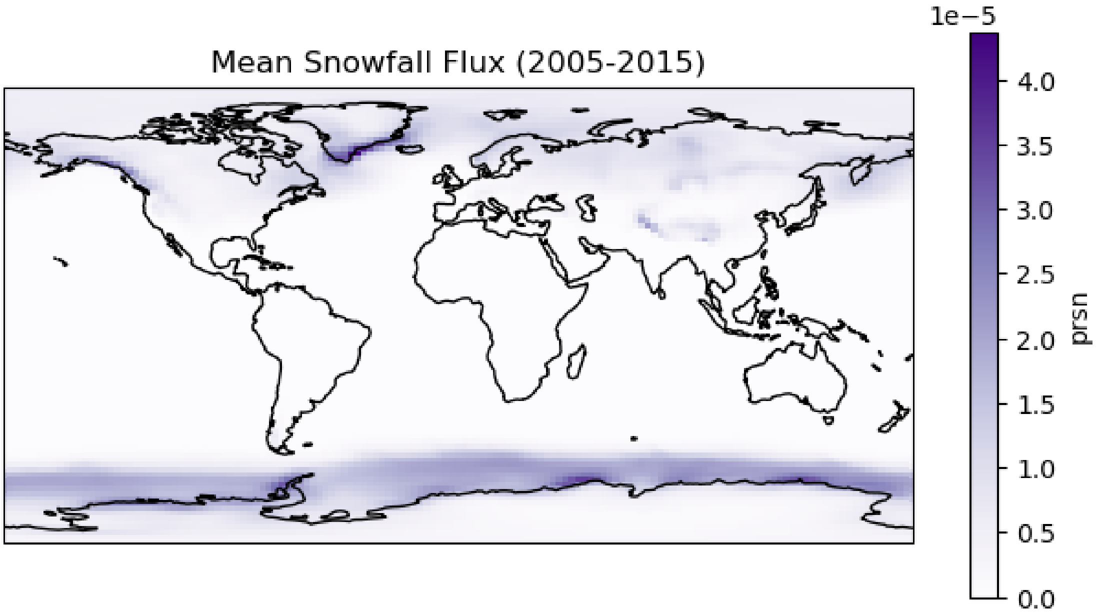
Snowfall Flux

[29]:

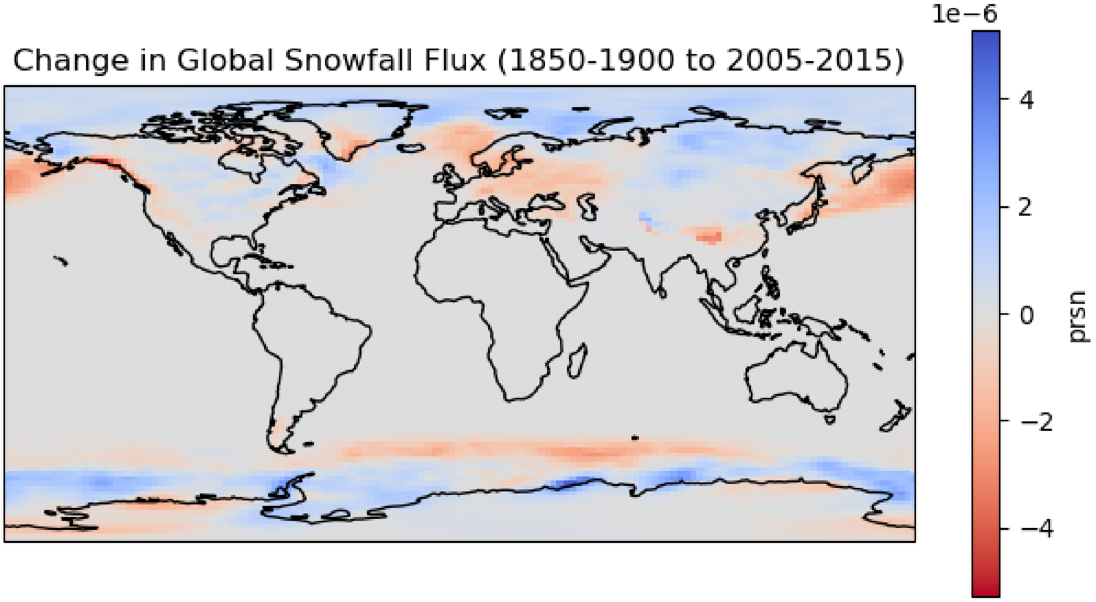
```
prsn = get_data('prsn', 'historical', 'r1i1p1f1')
```

[30]:

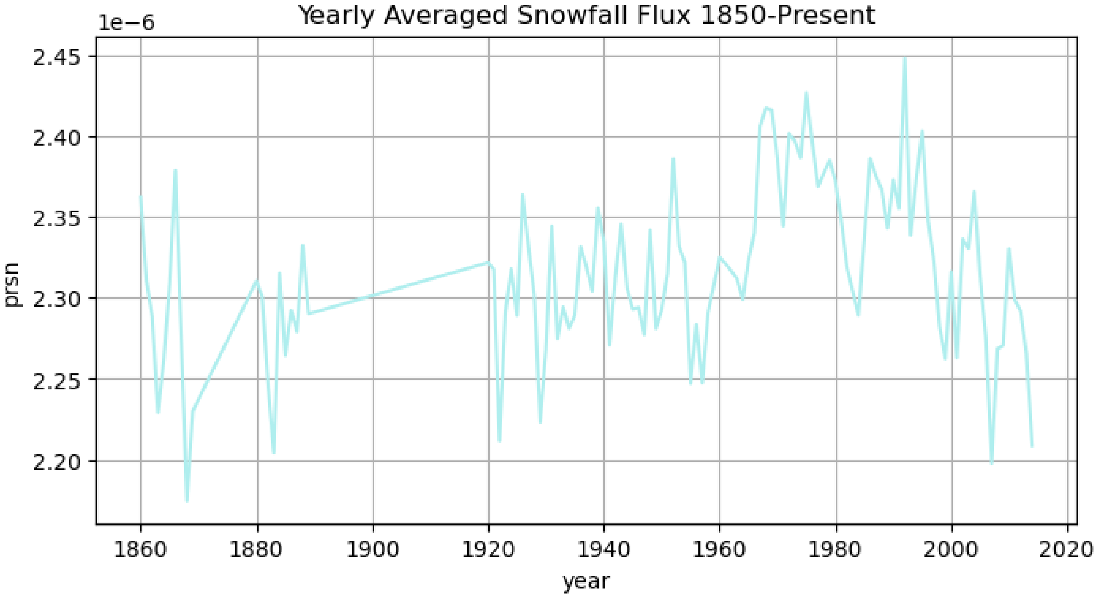
```
prsn.sel(time=slice('2005','2015')).mean('time').plot(
    transform=ccrs.PlateCarree(),
    subplot_kws={"projection": ccrs.PlateCarree()},
    cmap='Purples'
)

plt.title('Mean Snowfall Flux (2005-2015)')
plt.gca().coastlines()

plt.show()
```



```
[31]: (prsn.sel(time=slice('2005','2015')).mean('time') - prsn.
       ↪sel(time=slice('1850','1900')).mean('time')).plot(
          transform=ccrs.PlateCarree(),
          subplot_kws={"projection": ccrs.PlateCarree()},
          cmap="coolwarm_r",
      )

      plt.title('Change in Global Snowfall Flux (1850-1900 to 2005-2015)')
      plt.gca().coastlines()

      plt.show()
```



[32]:
```
prsn_timeseries = prsn.weighted(weights).mean(['lat', 'lon'])
yearly_prsn = prsn_timeseries.groupby("time.year").mean()

yearly_prsn.plot(color='paleturquoise')

plt.grid()
plt.title('Yearly Averaged Snowfall Flux 1850-Present')

plt.show()
```



[ ]: