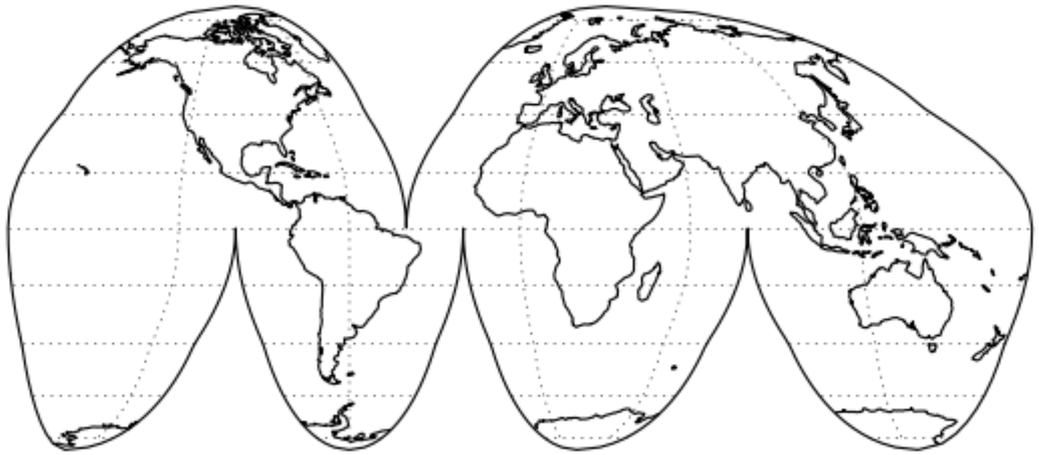
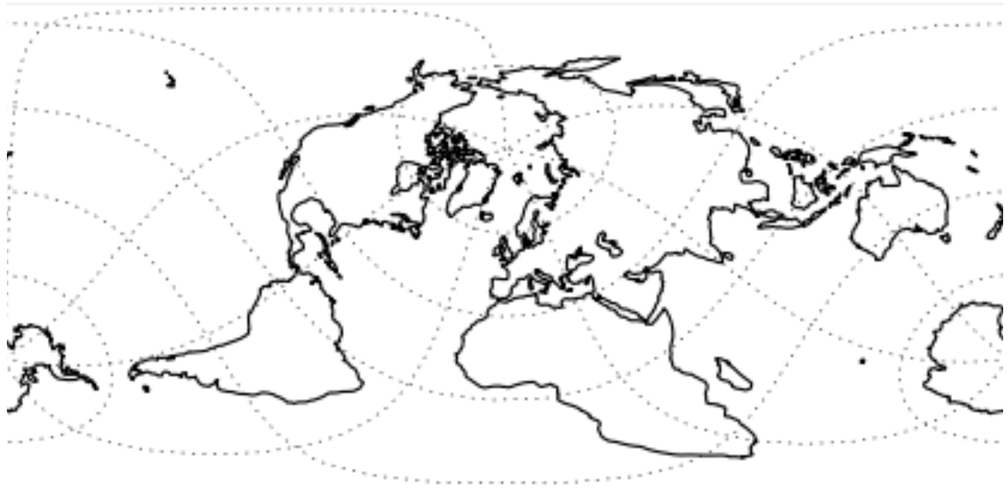




Royal Netherlands
Meteorological Institute
Ministry of Infrastructure
and Water Management

Plotting maps with Python + Basemap / Cartopy

Bart van Stratum

Introduction

- Basemap: was the standard for plotting maps
  - *Developed by Jeffrey Whitaker (NOAA)*
  - *Support until ~2020 (EOL together with Python 2.7)*
    - *"All new software development should try to use Cartopy whenever possible, and existing software should start the process of switching over to use Cartopy."*
- Cartopy: is/should become the new standard…
  - *Developed by UK MetOffice*
  - *Good basis, but lacks some of Basemap's features / stability*

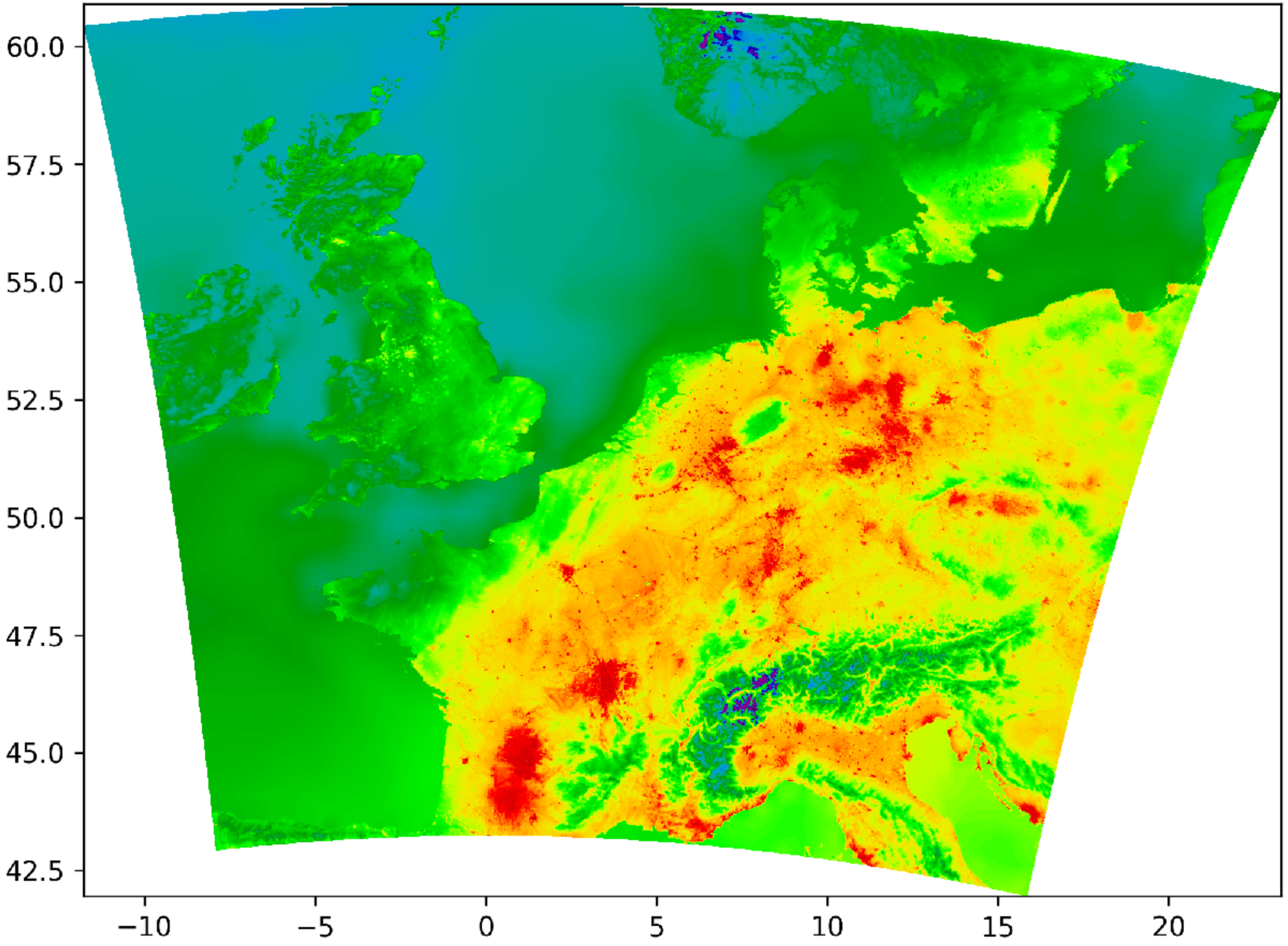
Outline talk

- Not an overview of every single Basemap / Cartopy option
  - *https://matplotlib.org/basemap/* & *http://scitools.org.uk/cartopy/*

- Some practical examples covering the basics
  - *From unprojected plot to Basemap / Cartopy*
  - *Some often used options (add country outlines, …)*

No map projection

```
import numpy as np
import matplotlib.pyplot as pl
import netCDF4 as nc4

# Example data (surface temperature Harmonie)
f   = nc4.Dataset('ts.his.NETHERLANDS.DOWA_40h12tg2_fERA5_master.20160623.nc')
lat = f.variables['lat'][:,:]
lon = f.variables['lon'][:,:]
ts  = f.variables['ts'][18]

# Plot without any projection
pl.figure(figsize=(8,8))
pl.pcolormesh(lon, lat, ts, cmap=pl.cm.nipy_spectral)
```

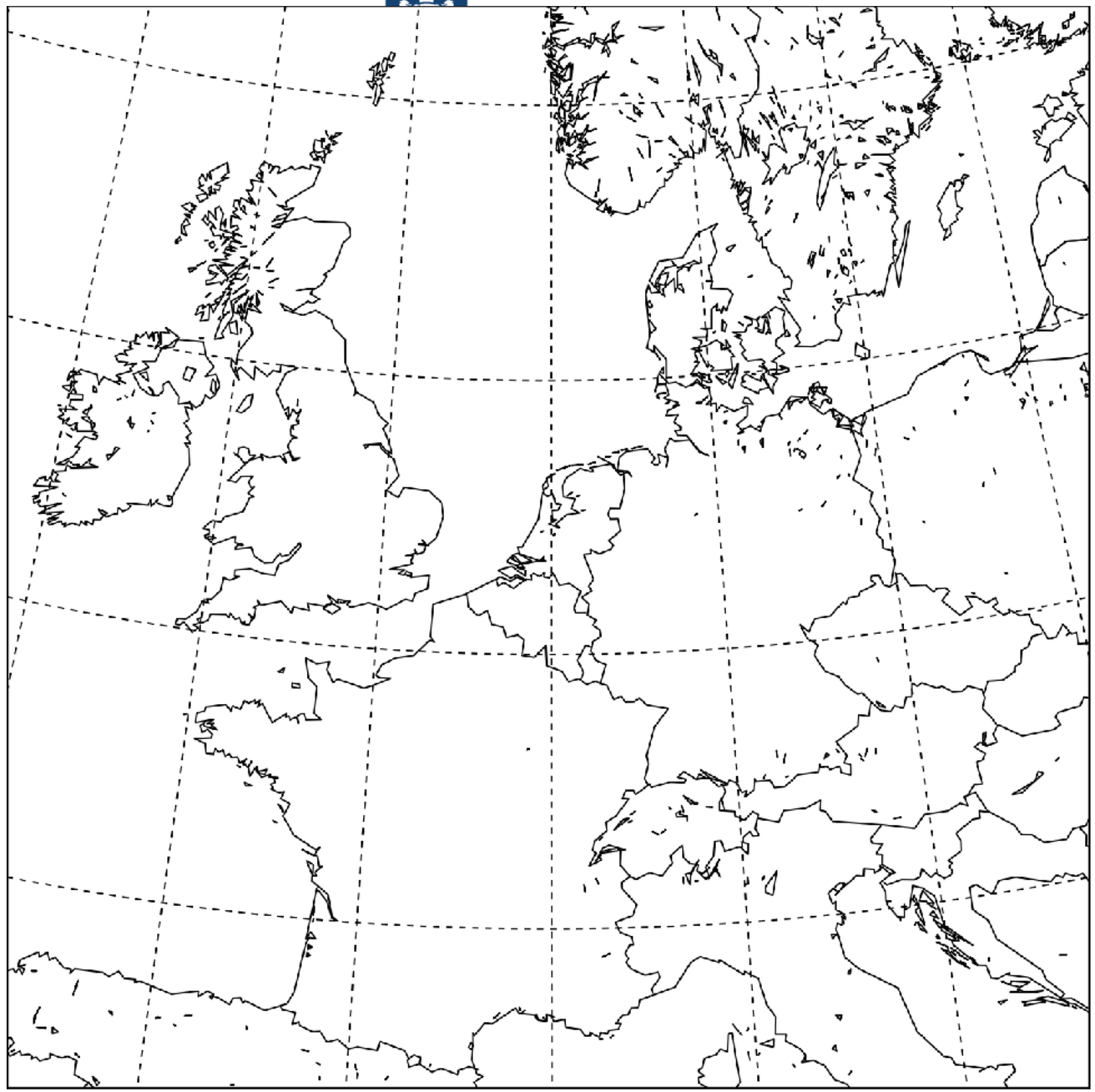

60.0
57.5
55.0
52.5
50.0
47.5
45.0
42.5
−10
−5
0
5
10
15
20

Setup map (Lambert Conformal) with Basemap

```
from mpl_toolkits.basemap import Basemap

pl.figure(figsize=(8,6))

# Setup Basemap (arguments depend on projection…)
m = Basemap(width=2200000, height=2200000,
            rsphere=(6378137.00,6356752.3142),\
            resolution='l', area_thresh=10., projection='lcc',\
            lat_0=51.97, lon_0=4.9)

m.drawparallels(np.arange(-80.,81.,5.), linewidth=0.5)
m.drawmeridians(np.arange(-180.,181.,5.), linewidth=0.5)
m.drawcoastlines(linewidth=0.5)
m.drawcountries(linewidth=0.5)
```

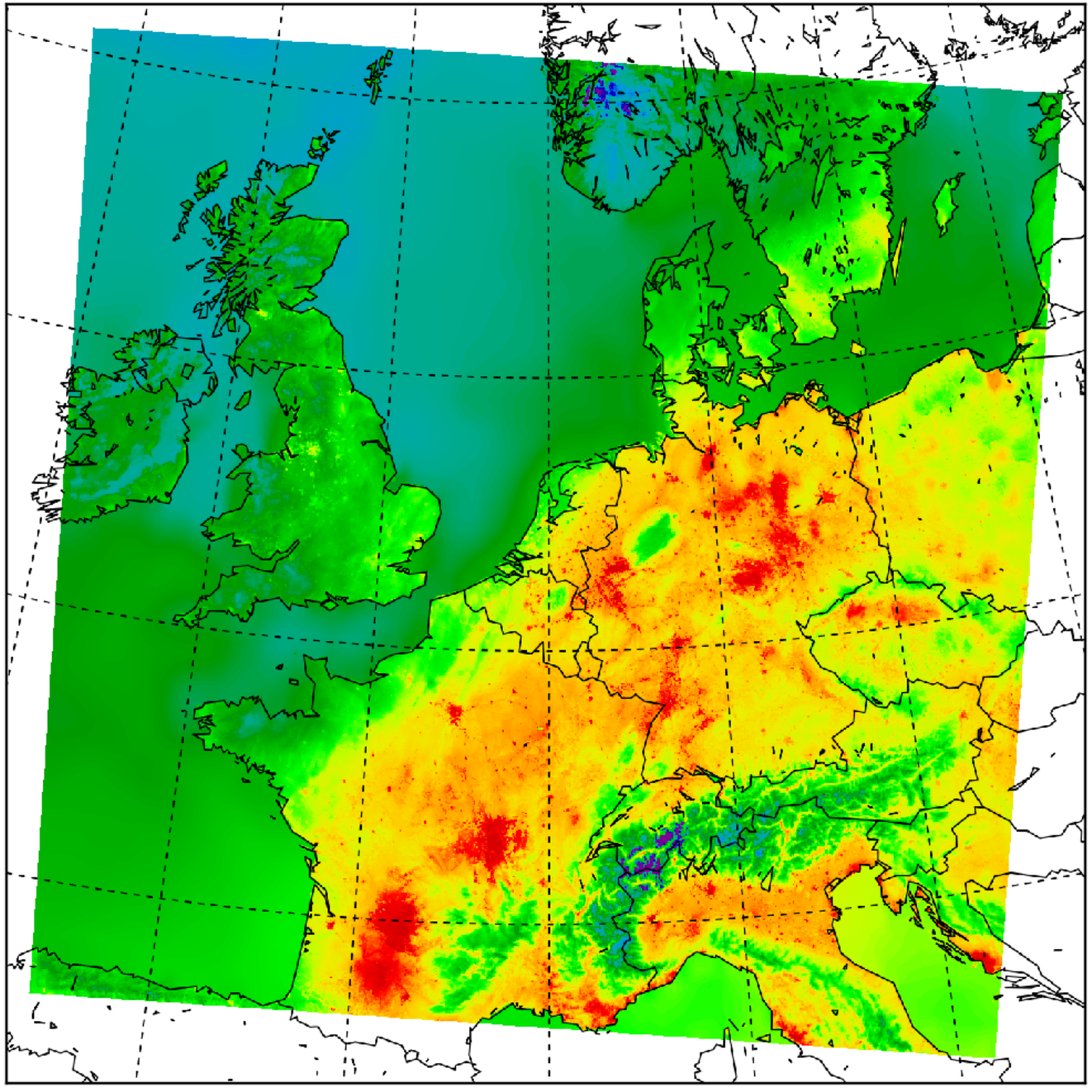
Result

- Figure coordinates in meters!



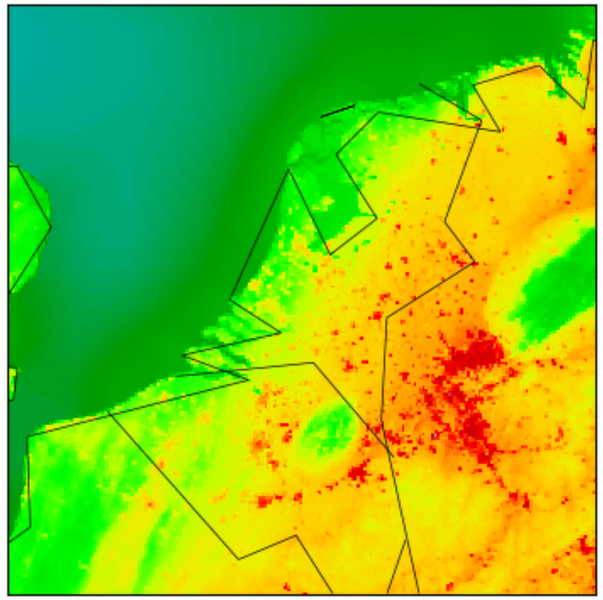
Data transform from lat/lon to projection

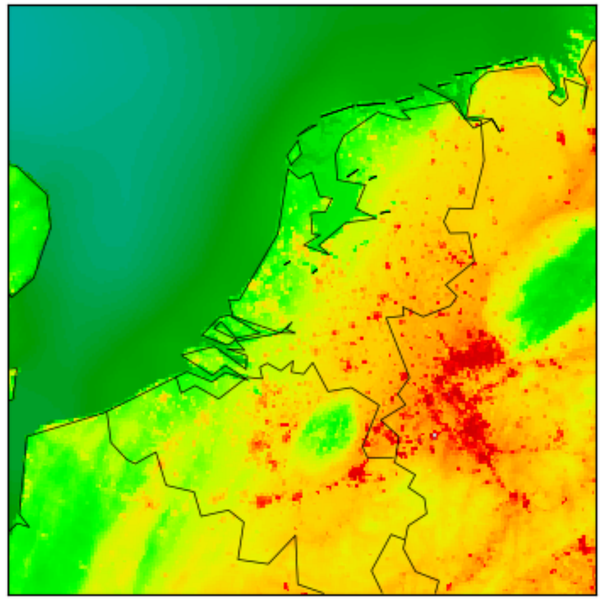
```
# Setup Basemap
m = Basemap(width=2200000,height=2200000,
            rsphere=(6378137.00,6356752.3142),\
            resolution='l', area_thresh=10., projection='lcc',\
            lat_0=51.97, lon_0=4.9)

# Option one (manual coordinate transform, use normal matplotlib routines)
x, y = m(lon, lat)
pl.pcolormesh(x, y, ts, cmap=pl.cm.nipy_spectral)

# Option two (specify latlon=True, use Basemap plot routines)
m.pcolormesh(lon, lat, ts, latlon=True, cmap=pl.cm.nipy_spectral)

m.drawparallels(np.arange(-80.,81.,5.), linewidth=0.5)
m.drawmeridians(np.arange(-180.,181.,5.), linewidth=0.5)
m.drawcoastlines(linewidth=0.5)
m.drawcountries(linewidth=0.5)
```

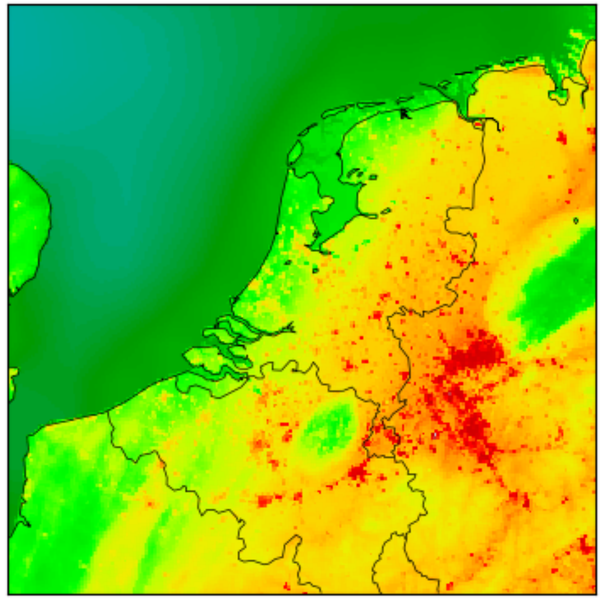


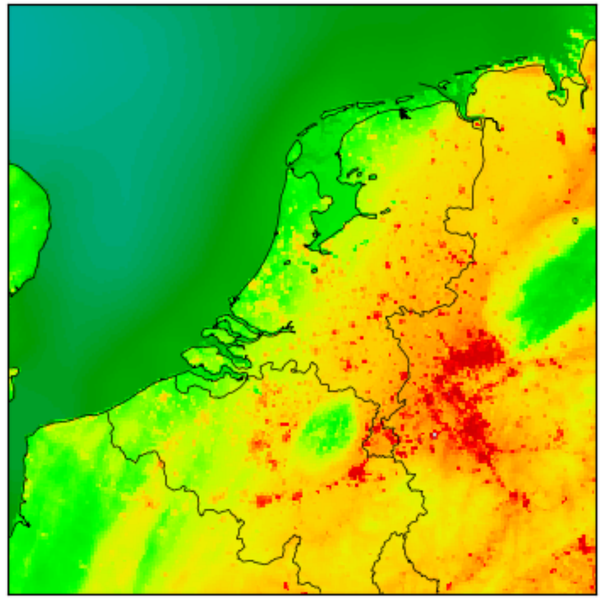
resolution = c, l, i, h

h = another coffee break





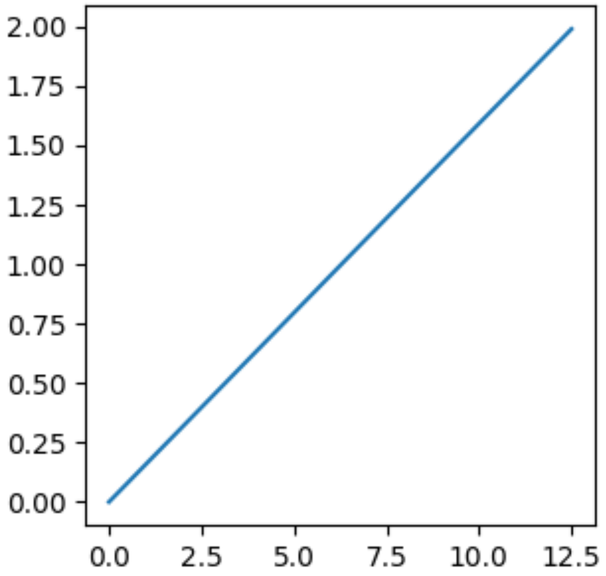


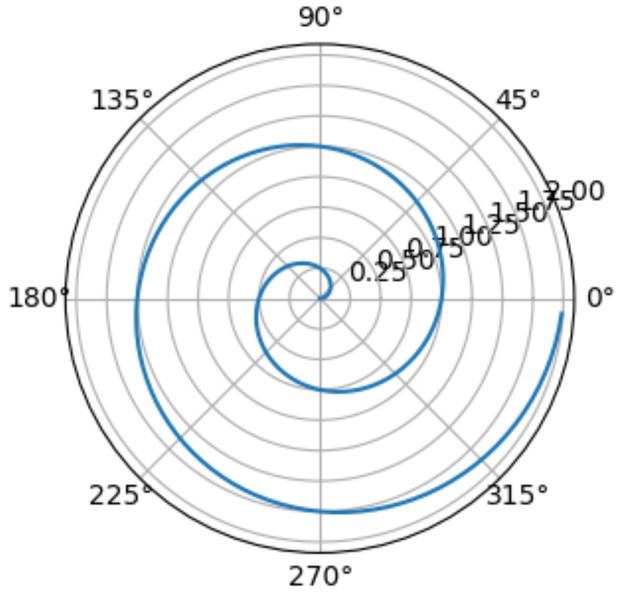


Cartopy

- Matplotlib core supports coordinate transforms/projections
- Cartopy built on top of these transforms

```
r = np.arange(0, 2, 0.01)
theta = 2 * np.pi * r

ax = pl.subplot(121)
ax.plot(theta, r)

ax = pl.subplot(122, projection='polar')
ax.plot(theta, r)
```



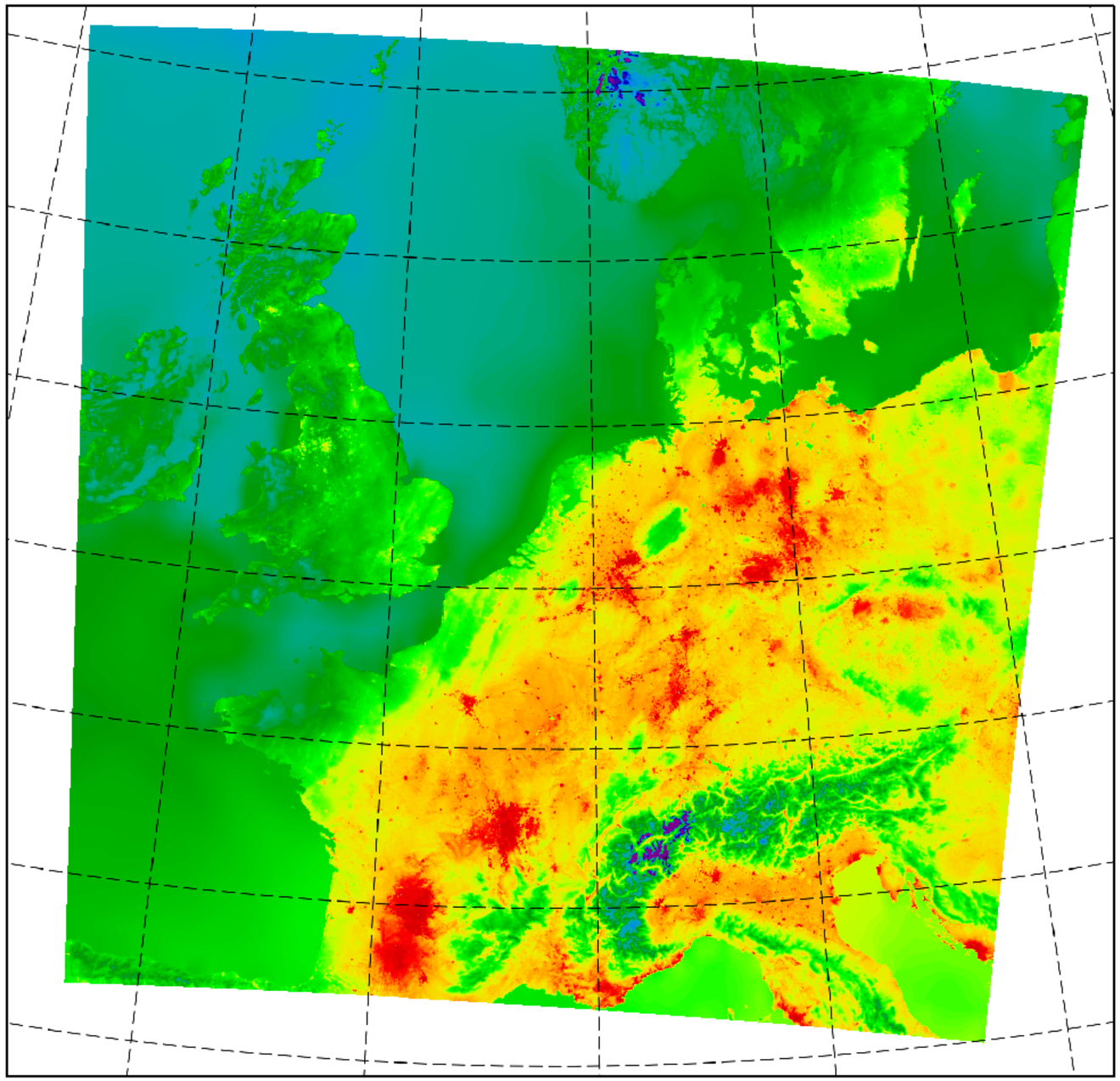


Setup map (Lambert Conformal) with Cartopy

```
import cartopy                          # Cartopy base
import cartopy.crs as ccrs              # Coordinate reference systems and transformations
import cartopy.feature as cfeature      # Features like coast lines, ...

pl.figure(figsize=(10,8))

# Create axes object in Lambert projection
lcc = ccrs.LambertConformal(central_longitude=4.9, central_latitude=51.967)
ax = pl.subplot(1,1,1, projection=lcc)

# Plot, specifying input coordinate system with `transform`
pl.pcolormesh(lon, lat, ts, cmap=pl.cm.nipy_spectral, transform=ccrs.PlateCarree())

# Axes coordinates are in the original coordinate system:
ax.set_extent([-9, 19, 41, 61])

# Parallels / meridians
ax.gridlines(linestyle='–', linewidth=0.5, color='k')
```

Result

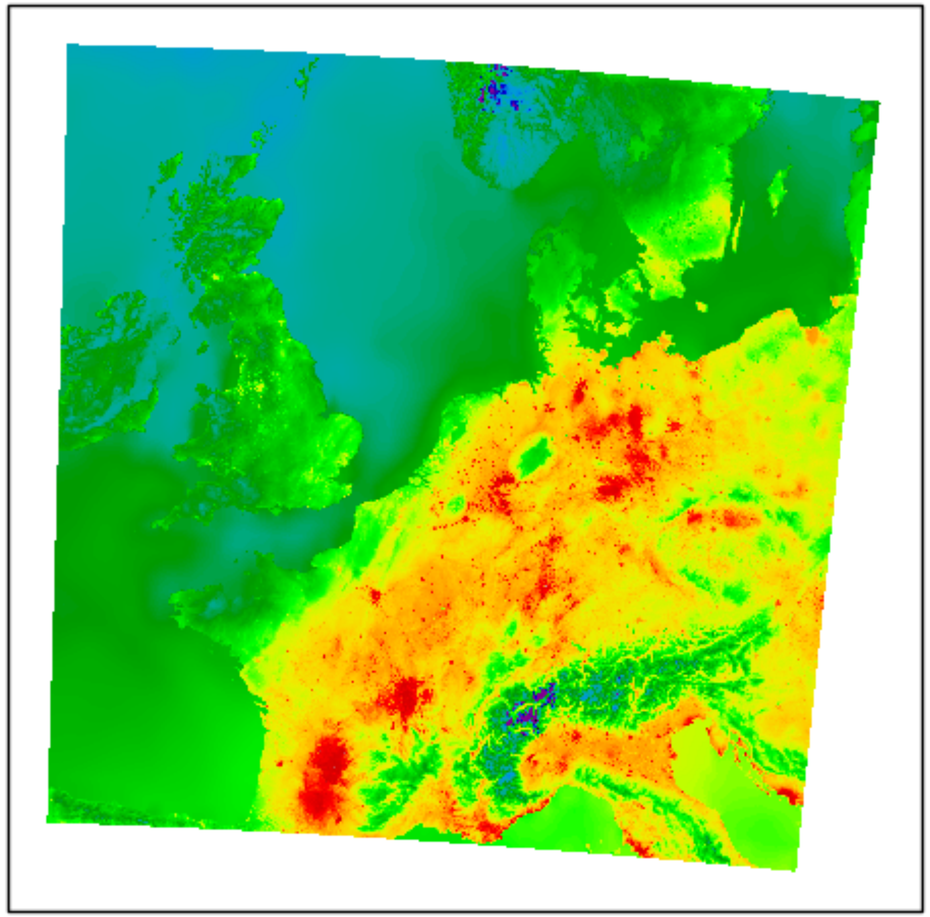
- Trapezoidal shape can be fixed by specifying the standard parallels

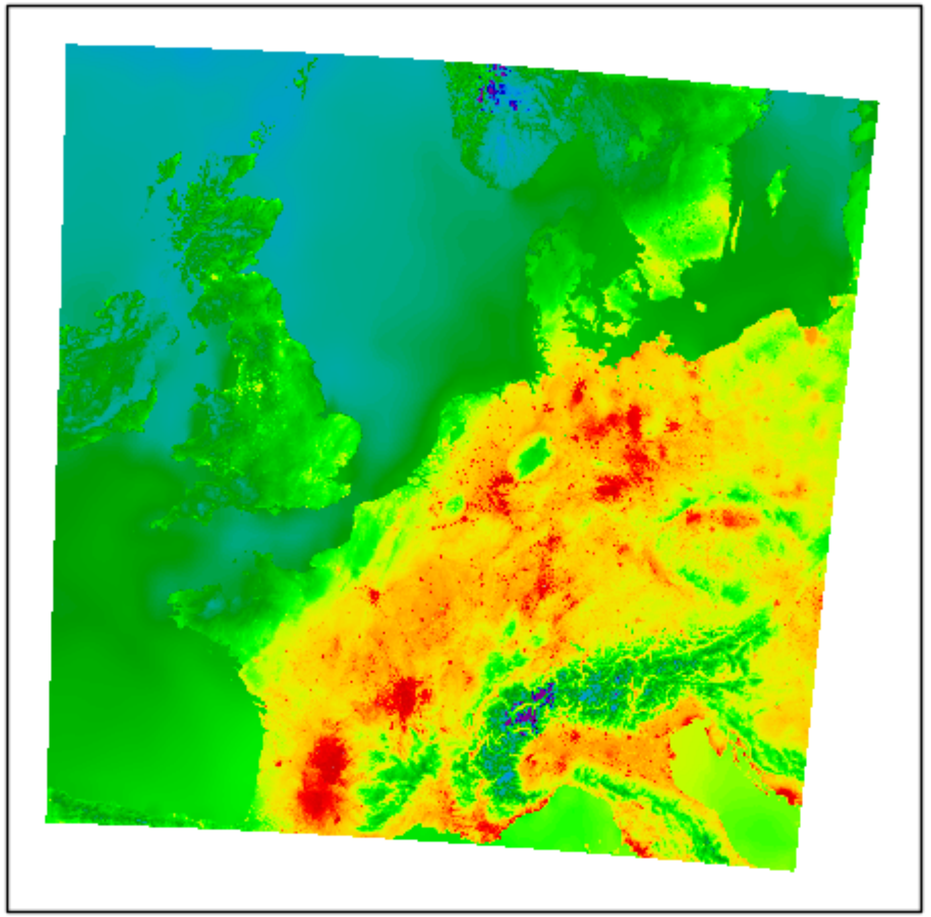


Projection & transform

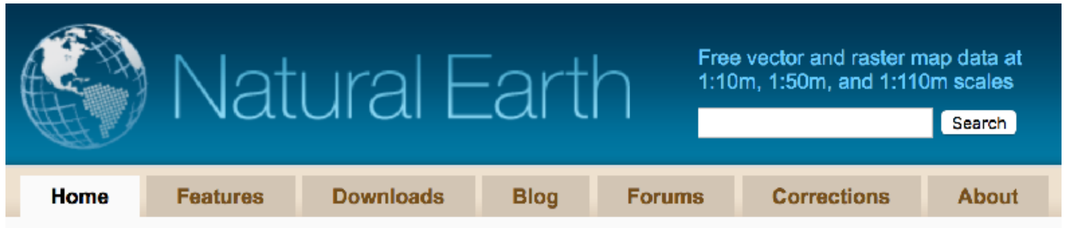
```
# Lambert & Albers projections
lcc = ccrs.LambertConformal(central_longitude=4.9, central_latitude=51.967)
aea = ccrs.AlbersEqualArea()

# "normal" method
ax = pl.subplot(121, projection=lcc)
pl.pcolormesh(lon, lat, ts, cmap=pl.cm.nipy_spectral, transform=ccrs.PlateCarree())

# Lets say some model outputs the coordinates in a Albers Equal Area projection,
# here mimicked offline with a transform in Python:
coords = aea.transform_points(ccrs.PlateCarree(), lon, lat)
x = coords[:,:,0]
y = coords[:,:,1]

ax = pl.subplot(122, projection=lcc)
pl.pcolormesh(x, y, ts, cmap=pl.cm.nipy_spectral, transform=aea)
```

Projection & transform





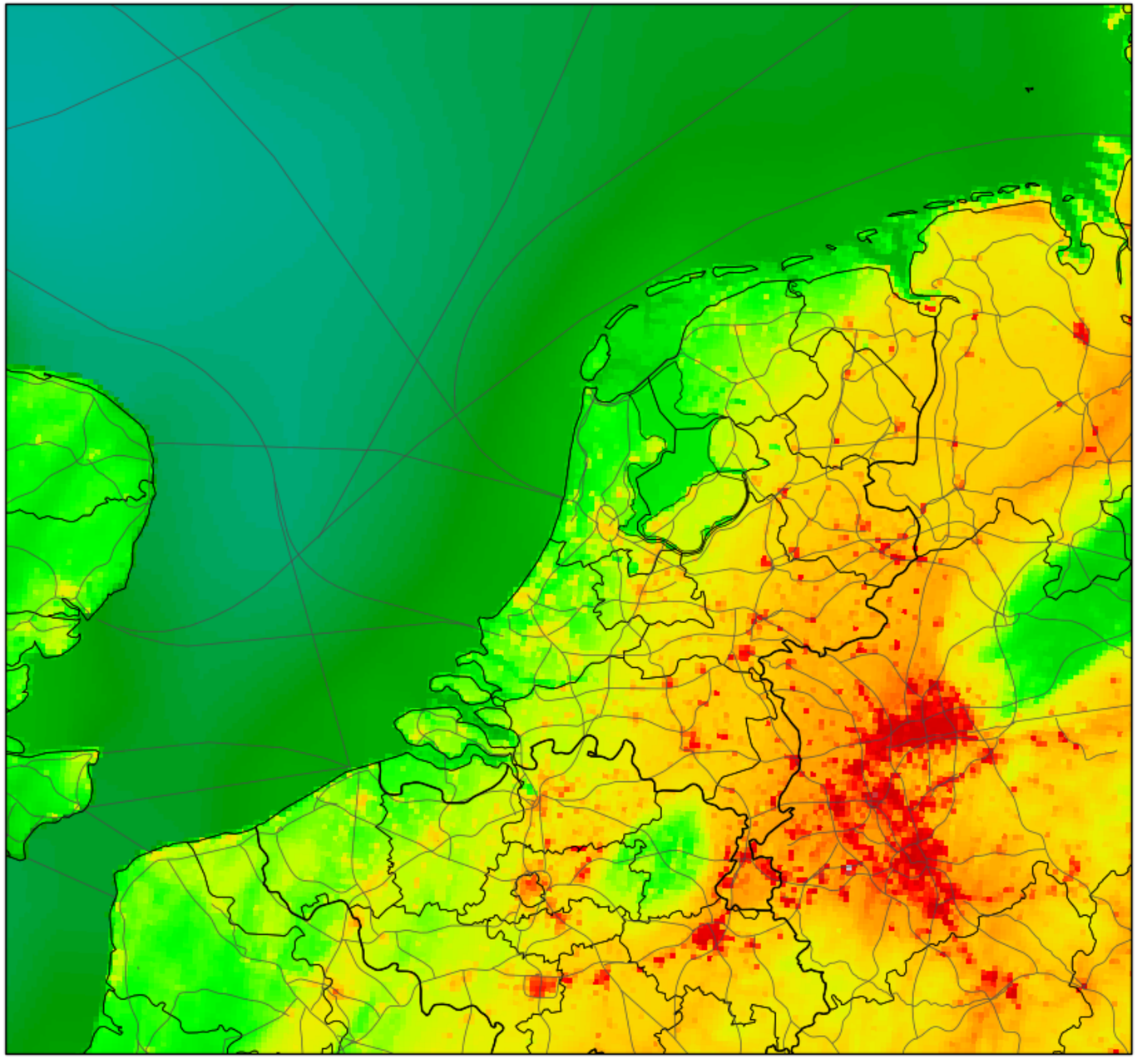
Adding map features



- Direct interface to naturalearthdata.com
  - *ESRI shapefile format*
  - *Country outlines, provinces, lakes/rivers/…, roads, …*

Adding map features

```
def add_feature(name, category, color='k', linewidth=1, scale='10m'):
    ax = pl.gca() # Get current axes object
    feature = cfeature.NaturalEarthFeature(
        category=category, name=name, scale=scale, facecolor='none')
    ax.add_feature(feature, edgecolor=color, linewidth=linewidth)

pl.figure(figsize=(10,8))

# Create single axes object in Lambert projection
lcc = ccrs.LambertConformal(central_longitude=4.9, central_latitude=51.967)
ax = pl.axes(projection=lcc)

add_feature('admin_1_states_provinces_scale_rank', 'cultural', linewidth=0.5, color='k')
add_feature('roads', 'cultural', color='0.3', linewidth=0.5)
add_feature('lakes', 'physical', linewidth=0.5)
add_feature('coastline', 'physical', linewidth=0.5)
add_feature('admin_0_boundary_lines_land', 'cultural')
```



Basemap or Cartopy?

- Cartopy
  - *Better core, cleaner integration in Matplotlib*
  - *Actively developed / maintained* *(thus sometimes incomplete and/or buggy)*
- Basemap
  - *More complete / well proven, but EOL in 2020….*

I'll put the code examples + presentation on the PWG Github repository:
https://github.com/folmerkrikken/knmi-python