Plot additional geodata

- **underlay** e.g. terrain data from a Digital Elevation Model (DEM)
- **overlay** features such as administrative borders, rivers, catchments, rain gauges, cities, ...

Here, we create a map without radar data to concentrate on the other layers.

[1]:

```
import wradlib as wrl
import matplotlib.pyplot as pl
import warnings
warnings.filterwarnings('ignore')
try:
    get_ipython().magic("matplotlib inline")
except:
    pl.ion()
import numpy as np
# Some more matplotlib tools we will need...
import matplotlib.ticker as ticker
from matplotlib.colors import LogNorm
from mpl_toolkits.axes_grid1 import make_axes_locatable
import cartopy.crs as ccrs
import cartopy.feature as cfeature
```

Plotting a Digital Elevation Model (DEM)

We use a preprocessed geotiff which was created from **SRTM** data via `gdal`

```
gdalwarp -te 88. 20. 93. 27. srtm_54_07.tif srtm_55_07.tif
srtm_54_08.tif srtm_55_08.tif bangladesh.tif
```

Here we - read the DEM via wradlib.io.open_raster and extracted via wradlib.georef.extract_raster_dataset. - resample the data to a (lon/lat) grid with `spacing=0.005`.

Note: we organise the code in functions which we can re-use in this notebook.

[2]:

```
def plot_dem(ax):
    filename =
wrl.util.get_wradlib_data_file('geo/bangladesh.tif')
    ds = wrl.io.open_raster(filename)
    # pixel_spacing is in output units (lonlat)
    ds = wrl.georef.reproject_raster_dataset(ds,
spacing=0.005)
    rastervalues, rastercoords, proj =
wrl.georef.extract_raster_dataset(ds)
    # specify kwargs for plotting, using terrain colormap and
LogNorm
    dem = ax.pcolormesh(rastercoords[..., 0],
rastercoords[..., 1],
                        rastervalues, cmap=pl.cm.terrain,
norm=LogNorm(),
                        vmin=1, vmax=3000,
transform=ccrs.PlateCarree())
    # add colorbar and title
    # we use LogLocator for colorbar
    cb = pl.gcf().colorbar(dem,
ticks=ticker.LogLocator(subs=range(10)))
    cb.set_label('terrain height [m]')
```

[3]:

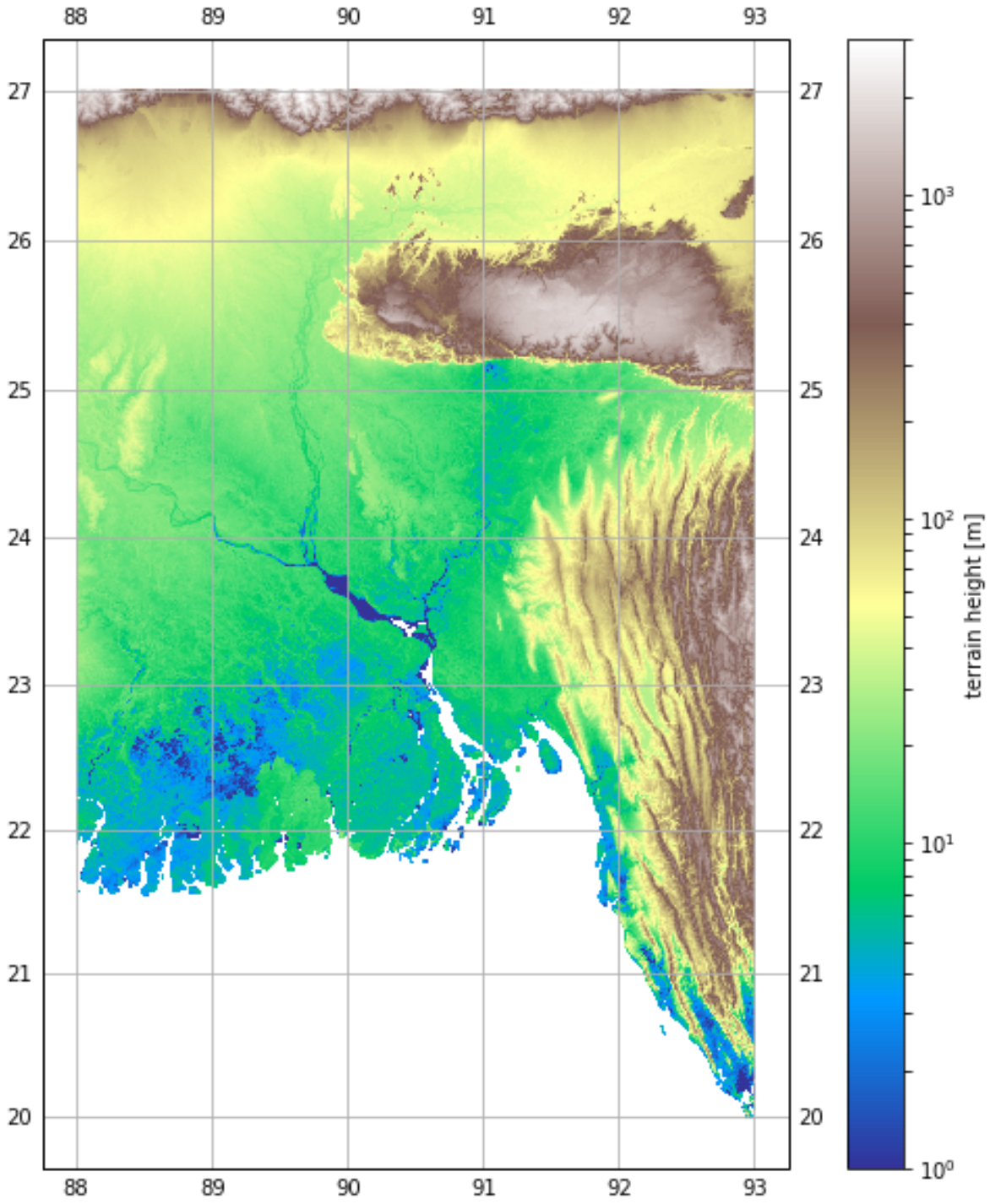
```
map_proj = ccrs.Mercator(central_longitude=90.5)
```

[4]:

```
fig = pl.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection=map_proj)
plot_dem(ax)
ax.gridlines(draw_labels=True)
```

[4]:

```
<cartopy.mpl.gridliner.Gridliner at 0x7f1205aee780>
```



Plotting borders

For country borders, we use ESRI Shapfiles from Natural Earth Data.

We extract features using - the `OGR.Layer` `AttributeFilter` and - the wradlib.georef.get_vector_coordinates function.

The plot overlay is done via wradlib.vis.add_lines.

[5]:

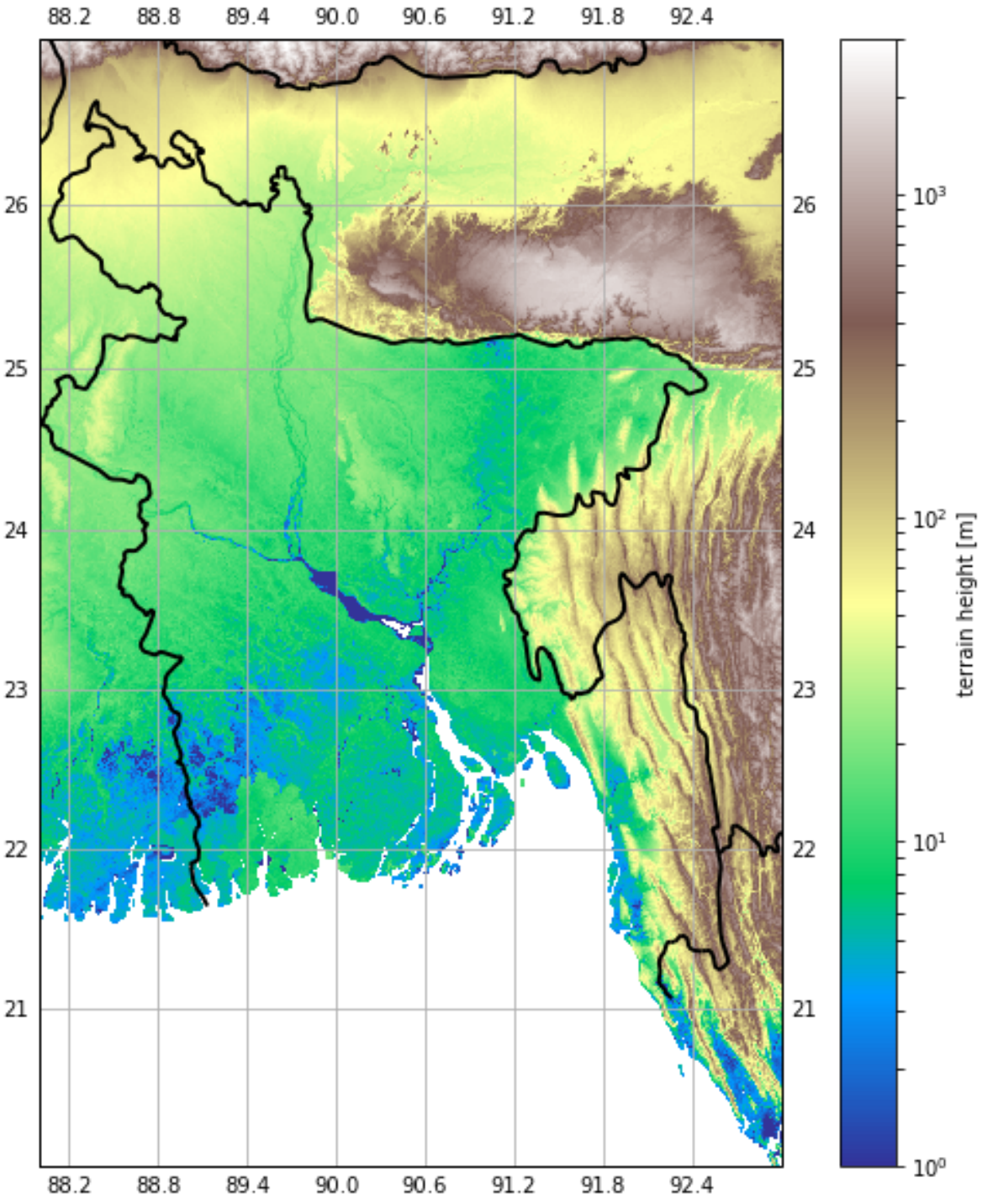
```
def plot_borders(ax):
    borders =
cfeature.NaturalEarthFeature(category='cultural',

name='admin_0_boundary_lines_land',
                                          scale='10m',
                                          facecolor='none')
    ax.add_feature(borders, edgecolor='black', lw=2,
zorder=4)
```

[6]:

```
fig = pl.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection=map_proj)
plot_dem(ax)
plot_borders(ax)
ax.set_extent([88, 93, 20, 27], crs=ccrs.PlateCarree())
ax.gridlines(draw_labels=True)
```

[6]:

```
<cartopy.mpl.gridliner.Gridliner at 0x7f1207394278>
```



Plotting Rivers

For rivers, we use ESRI Shapfiles from FAO and Natural Earth Data.

We extract features using - the `OGR.Layer` `SpatialFilter` and - the wradlib.georef.get_vector_coordinates function.

Then we use wradlib.vis.add_lines again for the overlay.

[7]:

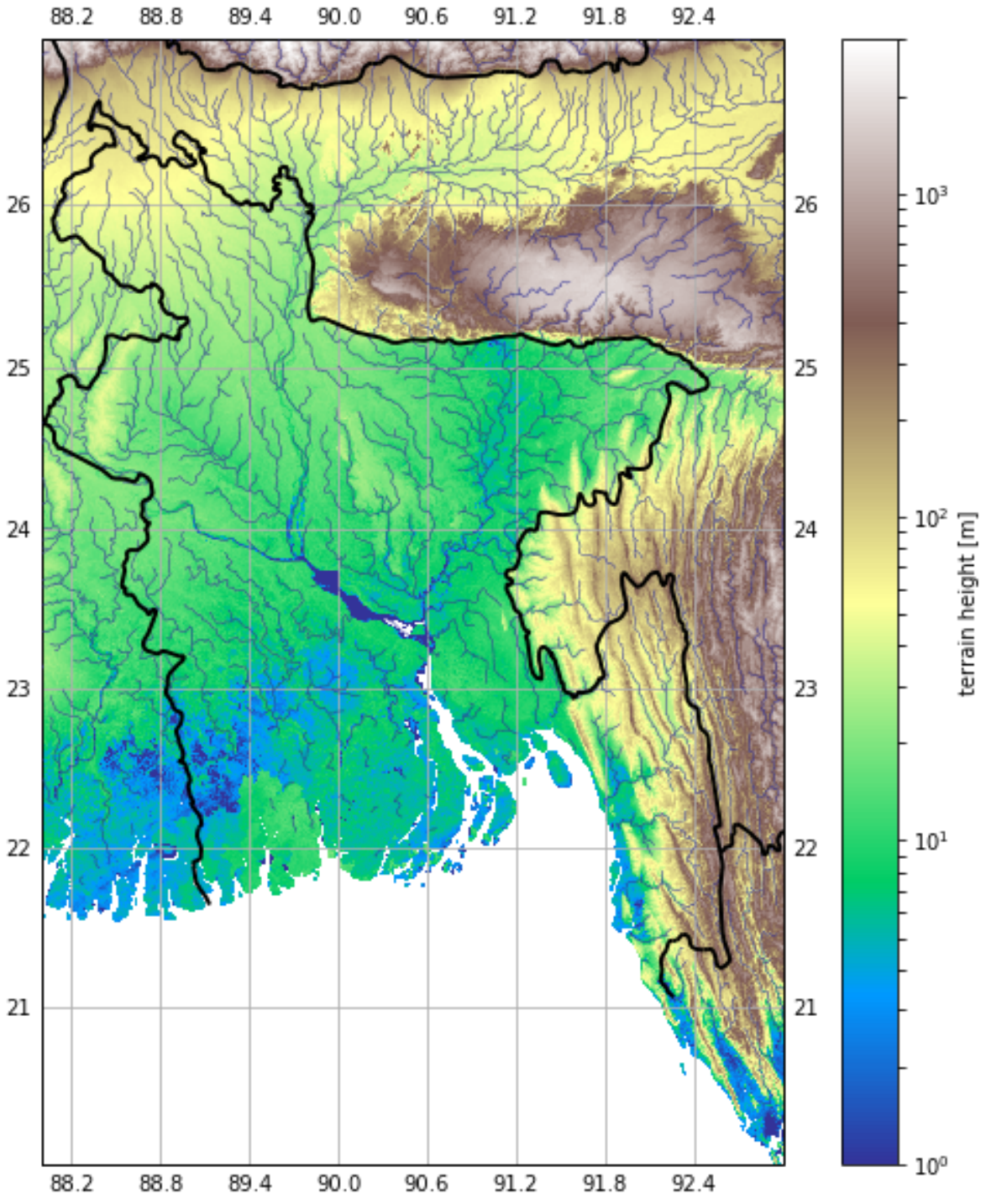
```
def plot_rivers(ax):
    # plot rivers from esri vector shape, filter spatially
    # http://www.fao.org/geonetwork/srv/en/metadata.show?
id=37331

    # open the input data source and get the layer
    filename =
wrl.util.get_wradlib_data_file('geo/rivers_asia_37331.shp')
    dataset, inLayer = wrl.io.open_vector(filename)

    # do spatial filtering to get only geometries inside
bounding box
    inLayer.SetSpatialFilterRect(88, 20, 93, 27)
    rivers, keys = wrl.georef.get_vector_coordinates(inLayer,
key='MAJ_NAME')

    # plot on ax1, and ax4
    wrl.vis.add_lines(ax, rivers, color=pl.cm.terrain(0.),
lw=0.5, zorder=3,
                      transform=ccrs.PlateCarree())
```

[8]:

```
fig = pl.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection=map_proj)
plot_dem(ax)
plot_borders(ax)
plot_rivers(ax)
ax.set_extent([88, 93, 20, 27], crs=ccrs.PlateCarree())
ax.gridlines(draw_labels=True)
```

[8]:

```
<cartopy.mpl.gridliner.Gridliner at 0x7f12008d9128>
```


88.2
88.8
89.4
90.0
90.6
91.2
91.8
92.4
26
25
24
23
22
21
10^3
10^2
10^1
10^0
terrain height [m]

[9]:

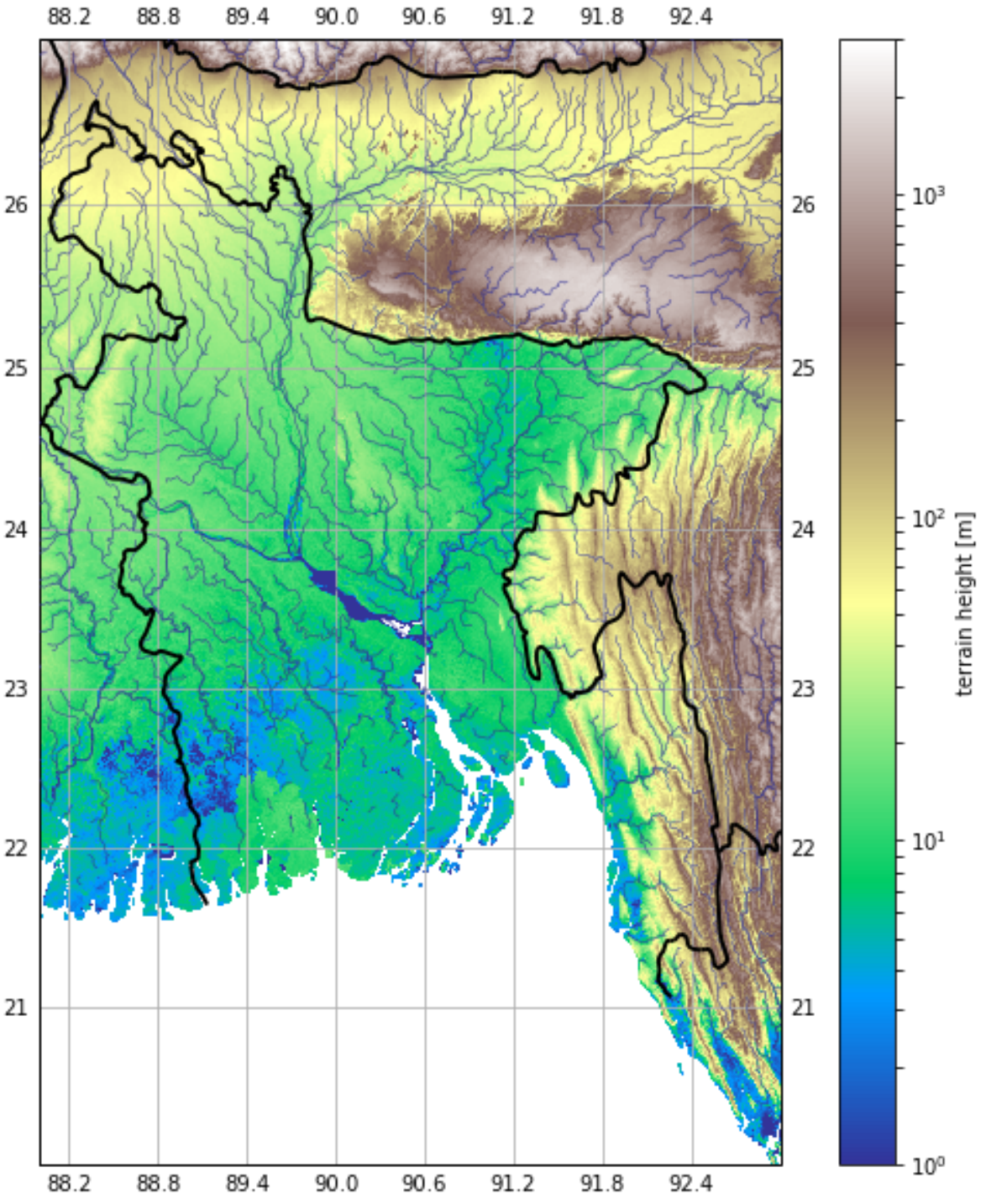
```
def plot_water(ax):
    # plot rivers from esri vector shape, filter spatially
    # plot rivers from NED
    # open the input data source and get the layer
    filename =
wrl.util.get_wradlib_data_file('geo/ne_10m_rivers_lake_'

'centerlines.shp')
    dataset, inLayer = wrl.io.open_vector(filename)
    inLayer.SetSpatialFilterRect(88, 20, 93, 27)
    rivers, keys = wrl.georef.get_vector_coordinates(inLayer)
    wrl.vis.add_lines(ax, rivers, color=pl.cm.terrain(0.),
lw=0.5, zorder=3,
                      transform=ccrs.PlateCarree())
```

[10]:

```
fig = pl.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection=map_proj)
plot_dem(ax)
plot_borders(ax)
plot_rivers(ax)
plot_water(ax)
ax.set_extent([88, 93, 20, 27], crs=ccrs.PlateCarree())
ax.gridlines(draw_labels=True)
```

[10]:

```
<cartopy.mpl.gridliner.Gridliner at 0x7f11f656f470>
```



Plotting Cities

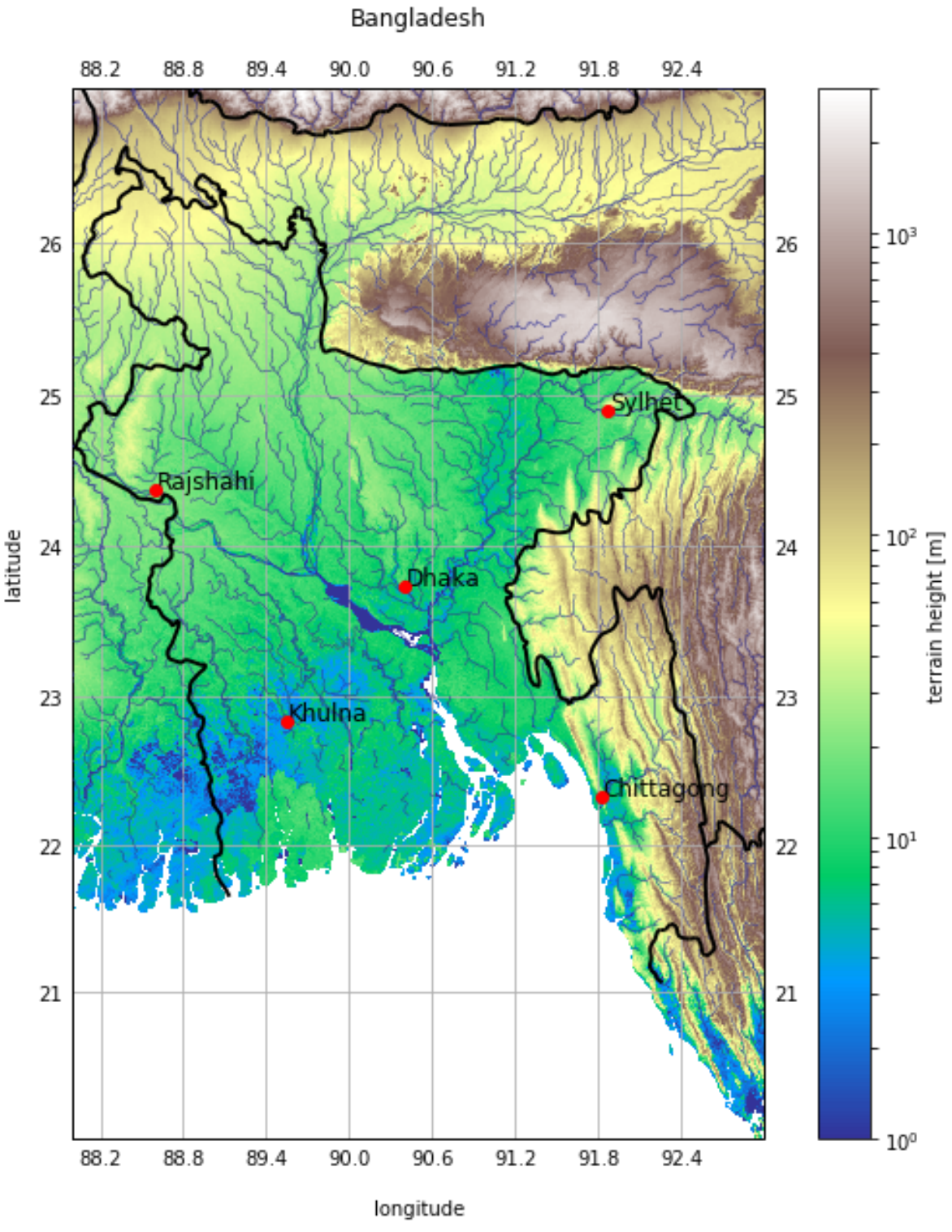
The 5 biggest cities of bangladesh are added using simple matplotlib functions.

[11]:

```
def plot_cities(ax):
    # plot city dots with annotation, finalize plot
    # lat/lon coordinates of five cities in Bangladesh
    lats = [23.73, 22.32, 22.83, 24.37, 24.90]
    lons = [90.40, 91.82, 89.55, 88.60, 91.87]
    cities = ['Dhaka', 'Chittagong', 'Khulna', 'Rajshahi',
'Sylhet']
    for lon, lat, city in zip(lons, lats, cities):
        ax.plot(lon, lat, 'ro', zorder=5,
transform=ccrs.PlateCarree())
        ax.text(lon + 0.01, lat + 0.01, city,
fontsize='large',
                transform=ccrs.PlateCarree())
```

[12]:

```
fig = pl.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection=map_proj)
plot_dem(ax)
plot_borders(ax)
plot_rivers(ax)
plot_water(ax)
plot_cities(ax)

ax.set_extent([88, 93, 20, 27], crs=ccrs.PlateCarree())
ax.gridlines(draw_labels=True)
ax.text(-0.07, 0.55, 'latitude', va='bottom', ha='center',
        rotation='vertical', rotation_mode='anchor',
        transform=ax.transAxes)
ax.text(0.5, -0.075, 'longitude', va='bottom', ha='center',
        rotation='horizontal', rotation_mode='anchor',
        transform=ax.transAxes)
t = ax.set_title('Bangladesh')
t.set_y(1.05)
```



Plotting country patches

Plotting in “geographic projection” (WGS84)

Here, we plot countries as `patches` on a lat/lon (WGS84) map (data from Natural Earth Data again).

We again extract the features using - the `OGR.Layer` `SpatialFilter` and - wradlib.georef.get_vector_coordinates.

Then the patches are added one by one via wradlib.vis.add_patches.

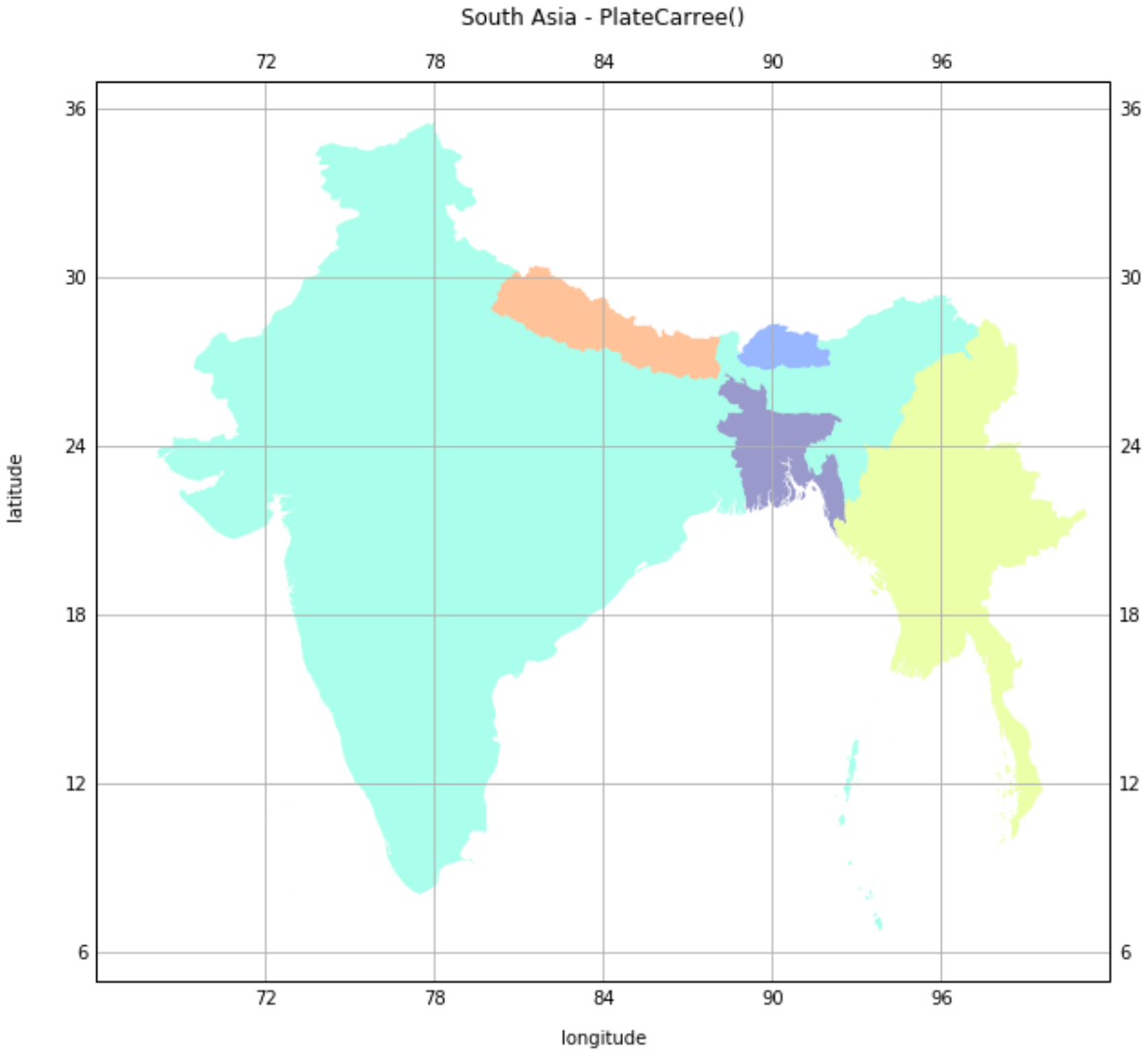
[13]:

```
def plot_wgs84(ax):
    from osgeo import osr
    wgs84 = osr.SpatialReference()
    wgs84.ImportFromEPSG(4326)
    # some testing on additional axes
    # add Bangladesh to countries
    countries = ['India', 'Nepal', 'Bhutan', 'Myanmar',
'Bangladesh']
    # create colors for country-patches
    cm = pl.cm.jet
    colors = []
    for i in range(len(countries)):
        colors.append(cm(1. * i / len(countries)))

    # open the input data source and get the layer
    filename =
wrl.util.get_wradlib_data_file('geo/ne_10m_admin_0_'

'countries.shp')
    dataset, layer = wrl.io.open_vector(filename)

    # filter spatially and plot as PatchCollection on ax3
    layer.SetSpatialFilterRect(88, 20, 93, 27)
    patches, keys = wrl.georef.get_vector_coordinates(layer,

dest_srs=wgs84,

key='name')
    i = 0
    for name, patch in zip(keys, patches):
        # why comes the US in here?
        if name in countries:
            wrl.vis.add_patches(ax, patch,
facecolor=colors[i],
                                cmap=pl.cm.viridis,
alpha=0.4,
                                transform=ccrs.PlateCarree())
            i += 1
    ax.autoscale(True)
    #ax.set_aspect('equal')
```

```
    #ax.set_xlabel('Longitude')
    #ax.set_ylabel('Latitude')
    ax.set_extent([66, 102, 5, 37], crs=ccrs.PlateCarree())
    ax.gridlines(draw_labels=True)
    ax.text(-0.07, 0.55, 'latitude', va='bottom',
ha='center',
            rotation='vertical', rotation_mode='anchor',
            transform=ax.transAxes)
    ax.text(0.5, -0.075, 'longitude', va='bottom',
ha='center',
            rotation='horizontal', rotation_mode='anchor',
            transform=ax.transAxes)
    t = ax.set_title('South Asia - PlateCarree()')
    t.set_y(1.05)
```

[14]:

```
fig = pl.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection=ccrs.PlateCarree())
plot_wgs84(ax)
```



Plotting with a map projection

Here, we plot countries as patches on a projected map.

We extract the features using - the `OGR.Layer` `AttributeFilter` and - the wradlib.georef.get_vector_coordinates function.

The coordinates of the features are reprojected on the fly using the `dest_srs` keyword of wradlib.georef.get_vector_coordinates.

Then, the patches are added to the map via wradlib.vis.add_patches.

[15]:

```
def plot_lambert(ax):
    # add Bangladesh to countries
    countries = ['India', 'Nepal', 'Bhutan', 'Myanmar',
'Bangladesh']
    # create colors for country-patches
    cm = pl.cm.jet
    colors = []
    for i in range(len(countries)):
        colors.append(cm(1. * i / len(countries)))

    # open the input data source and get the layer
    filename =
wrl.util.get_wradlib_data_file('geo/ne_10m_admin_0_'

'countries.shp')
    dataset, layer = wrl.io.open_vector(filename)
    # iterate over countries, filter by attribute,
    # plot single patches on ax2
    for i, item in enumerate(countries):
        fattr = "name = '" + item + "'"
        layer.SetAttributeFilter(fattr)
        # get country patches
        patches, keys =
wrl.georef.get_vector_coordinates(layer,

key='name')
        wrl.vis.add_patches(pl.gca(), patches,
facecolor=colors[i],
                            transform=ccrs.PlateCarree())

    ax.set_extent([66, 102, 5, 37], crs=ccrs.PlateCarree())
    ax.gridlines()
    ax.text(-0.07, 0.55, 'latitude', va='bottom',
ha='center',
            rotation='vertical', rotation_mode='anchor',
            transform=ax.transAxes)
    ax.text(0.5, -0.075, 'longitude', va='bottom',
ha='center',
            rotation='horizontal', rotation_mode='anchor',
            transform=ax.transAxes)
    t = ax.set_title('South Asia in
LambertAzimuthalEqualArea')
    t.set_y(1.05)
```

[16]:

```
fig = pl.figure(figsize=(10,10))
ax = fig.add_subplot(111,
projection=ccrs.LambertAzimuthalEqualArea(central_longitude=84


central_latitude=20))
plot_lambert(ax)
```