Equal-Area Map Projections with Basemap and matplotlib/pylab

Arne Babenhauserheide

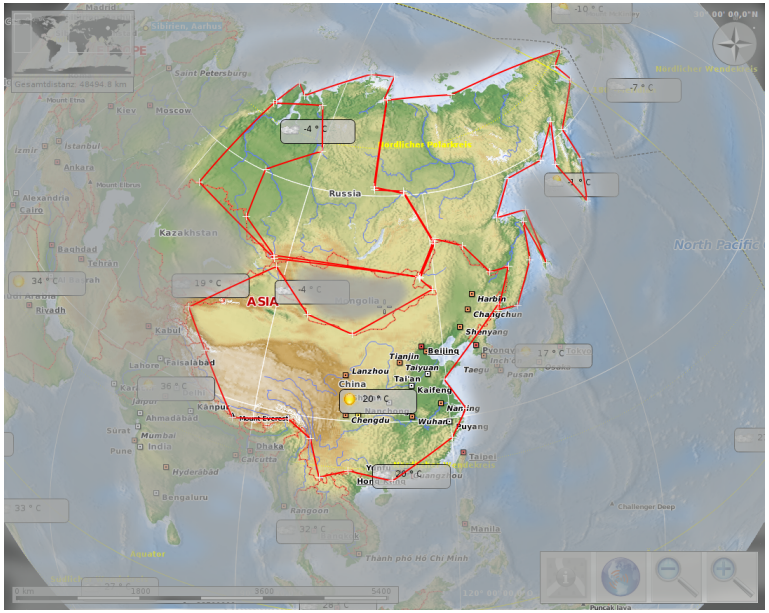
June 25, 2013

Outline

lat/lon pixels misrepresent areas

Simple Flat

Globe



Sibiria $13{,}1 \cdot 10^6$ km² vs. china $9.7 \cdot 10^6$ km²

Maps thanks to Marble Desktop Globe and Open Street Map, available under CC by-sa and Open Data Commons Open Database License (ODbL).

Map Projections

Hobo-Dyer

- Rectangle
- equidistant longitude
- longitude/latitude over the Mediterranean Sea (more exactly: 37.5°)
- Similar Maps: Gall-Peters (thinner), Lambert (wider)
- Basemap: equal area cylindrical (cea) with lat_{ts}=37.5

Hammer

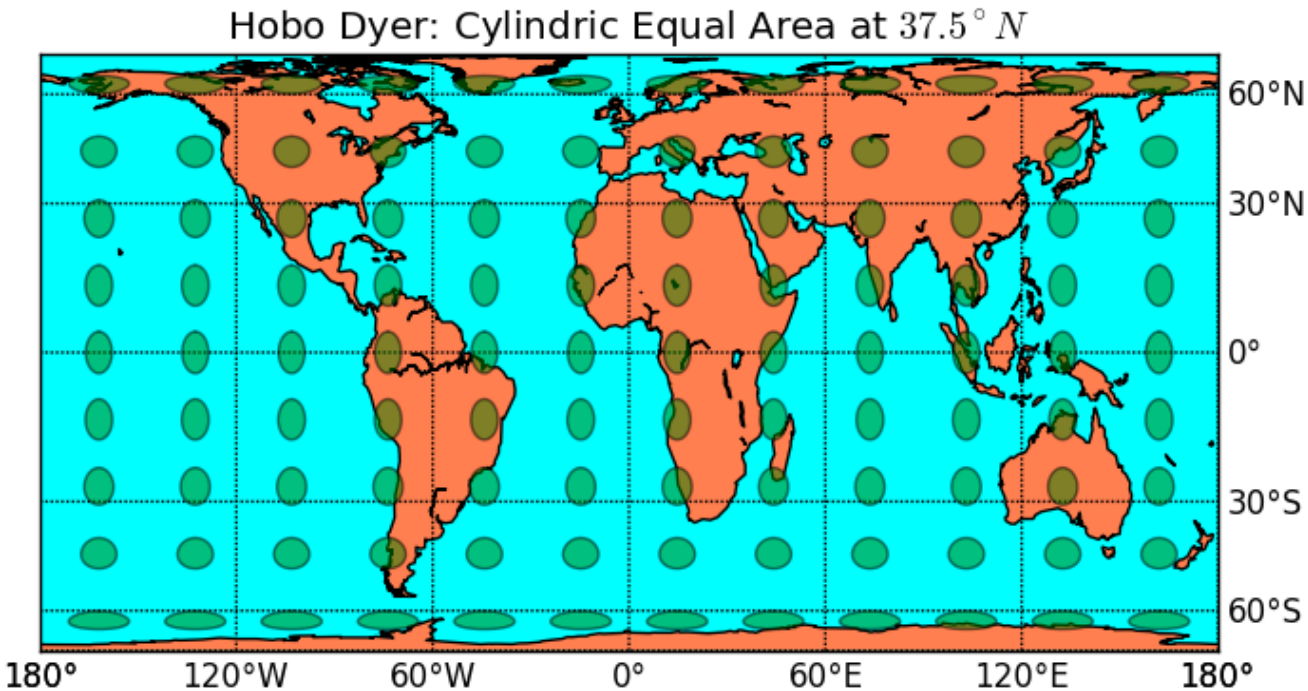
- Elliptic
- Low distortion at the poles
- 2:1 $\rightarrow$ 2 per page
- the earth appears round without making it hard to recognize patterns
- Similar Maps: Mollweide (more distorted at the poles, parallel latitudes)
- Basemap: hammer

Flat Polar Quartic

- Elliptic with polar cuts
- parallel latitudes
- Standard parallels at $33°45'N/S$
- poles are $\frac{1}{3}$ the equator
- Similar: Eckart IV (poles are half the equator)
- Basemap: mbtfpq

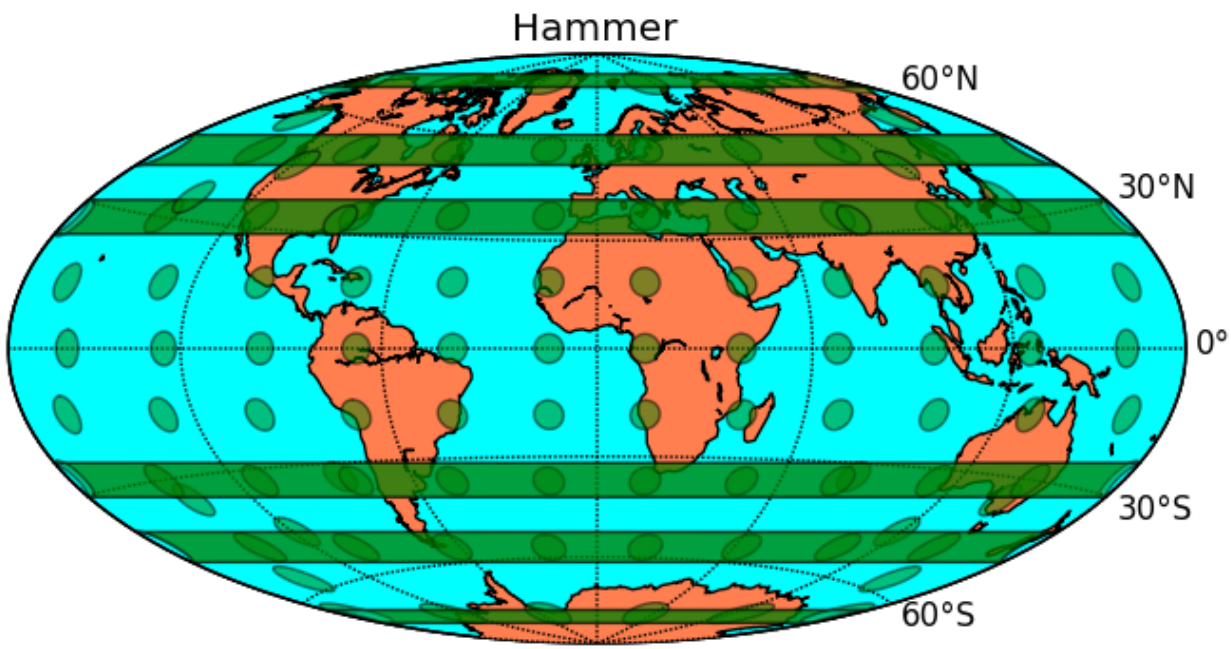
Hobo-Dyer

```
m = map.Basemap(projection='cea', lat_ts=37.5)
outfile = "hobo-dyer.png"
pl.title("Hobo Dyer: Cylindric Equal Area at $37.5^\\circ N$")
```



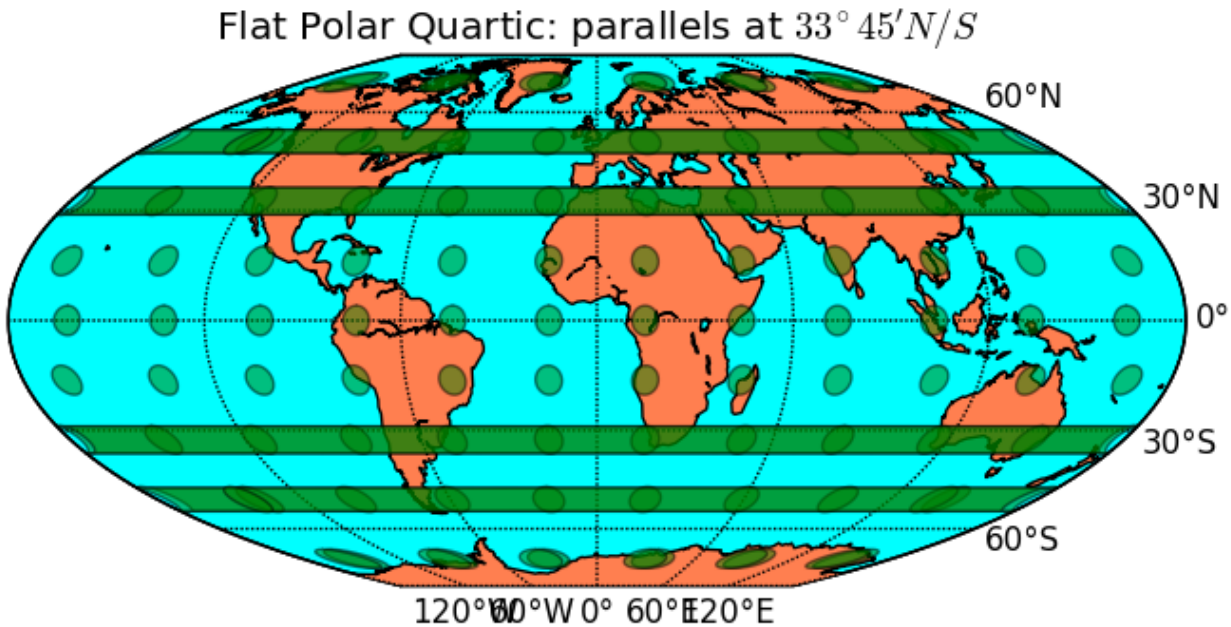
Hammer

```
m = map.Basemap(projection='hammer', lon_0=0)
outfile = "hammer.png"
pl.title("Hammer") # latex-test: $\frac{1}{2}$
```



Flat Polar Quartic

```
m = map.Basemap(projection='mbtfpq', lon_0=0)
outfile = "flatpolarquartic.png"
pl.title("Flat Polar Quartic: parallels at $33^\\circ 45' N/S$")
```

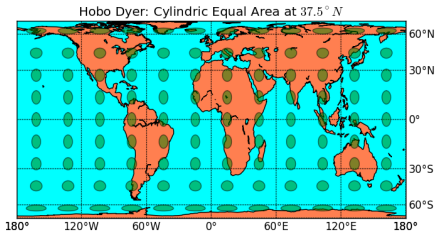


Other Equal Area map types

- Goode homolosine: Split, focus on land or ocean, straight latitude parallels, approximately preserve most shapes. Not available in matplotlib.
- Eckert IV: Like Flat Polar Quartic, parallels at 40° 30' N/S, poles are *half* the equator.
- Lambert cylindrical equal area: Like Hobo Dyer, very wide, shapes at the equator are correct.
- Gall Peters: Like Hobo Dyer, appears more distorted than Hobo-Dyer, shapes over europe correct (45°).
- Mollweide: Like Hammer with straight latitude parallels.
- Werner: It's a heart :) - focus on a hemisphere without ignoring the rest. General case: Bonne.
- Tobler: General case leading to Lambert, Mollweide, Mollington and a few more — also see Tobler1973 after you manage to gnawl through the paywall...
- Collignon: Triangle, for cosmic microwave background.

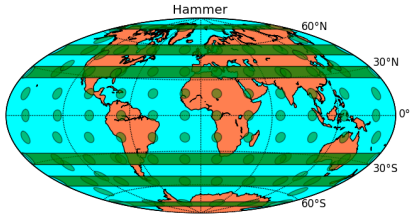
Maps I plan to use

Hobo-Dyer



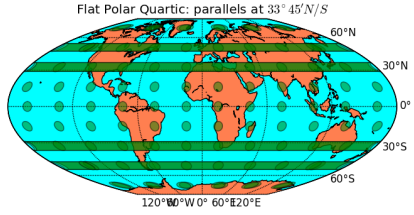
To show regional fluxes and longitudinally constrained regions: Easy to spot on rectangular grid.

Hammer



To show a global overview: Helps the understanding of global data because it appears most similar to a real earth while including the whole earth surface.

Flat Polar Quartic



For mainly latitudinally constrained regions: Straight latitudinal lines and high latitudinal resolution near the poles.

Thank you for listening!

Questions?

Basemap Imports

```
# basemap, pylab and numpy for plotting
import mpl_toolkits.basemap as map
import pylab as pl
import numpy as np
# netcdf for reading the emission files
import netCDF4 as nc
```

Listing 1: Basic imports for plotting with Basemap.

- Basemap: `http://matplotlib.org/basemap/`
- Pylab: `http://www.scipy.org/PyLab`
- Numpy: `http://www.numpy.org/`
- NetCDF: `http://code.google.com/p/netcdf4-python/`

Draw a map

```
<<addmapfeatures>>
<<addindicatrix>>
try:
  <<addemissions>>
  <<addcolorbar>>
except RuntimeError: # recover from missing fluxfile
  m.fillcontinents(color='coral',lake_color='aqua')
pl.savefig(outfile)
return "./" + outfile + ""
```

Listing 2: Commands to draw a map. If no emission data is available, use the default fill color for continents. Requires previous imports and a basemap setup as done in the plot snippets for the individual maps.

Map features

```
# add map lines
m.drawcoastlines()
# only fill continents if we do not plot emissions
# m.fillcontinents(color='coral',lake_color='aqua')
m.drawparallels(np.arange(-90.,120.,30.),
                labels=[False,True,True,False])
m.drawmeridians(np.arange(0.,420.,60.),
                labels=[True,False,False,True])
m.drawmapboundary(fill_color='aqua')
```

Listing 3: Commands to add visual orientation to a map.

Tissots Indicatrix

```
# draw tissot's indicatrix to show distortion.
for y in np.linspace(m.ymax/20,19*m.ymax/20,9):
    for x in np.linspace(m.xmax/20,19*m.xmax/20,12):
        lon, lat = m(x,y,inverse=True)
        poly = m.tissot(lon,lat,4.0,100,
                        facecolor='green',
                        zorder=10,alpha=0.5)
```

Listing 4: Add Tissots Indicatrix to visualize distortion.

Plot emissions

```
# d = nc.Dataset("/run/media/arne/3TiB/CTDAS-2013-03-07-2years-base-data/"
#                "analysis/data_flux1x1_weekly/flux_1x1.nc")
d = nc.Dataset("UNPUBLISHED")
biocovmean = np.mean(
    d.variables["bio_flux_prior_cov"][:,:,:], axis=0)
# projection: matplotlib.org/basemap/users/examples.html
lons, lats = pl.meshgrid(range(-180, 180),
                         range(-90, 90))
x, y = m(lons, lats)
# choose my standard color range: vmin = -0.5*vmax
vmax = max(abs(np.max(biocovmean)),
           2 * abs(np.min(biocovmean)))
vmin = -0.5*vmax
m.pcolor(x, y, biocovmean, shading='flat',
         vmin=vmin, vmax=vmax) # pcolormesh is faster
```

Listing 5: Plot Carbontracker emissions on a basemap (m).

Nice colorbar

```
pl.rcParams.update({"text.usetex": True,
                    "text.latex.unicode": True})
colorbar = pl.colorbar(orientation="horizontal",
                       format="%.2g") # scientific
colorbar.set_label("$CO_{2}$ fluxes [$\\frac{mol}{m^2 s}$]")
```

Listing 6: Add a nice colorbar to the plot.