Map Projections
The problem of forcing a spherical surface onto a map is an old problem in cartography. Spherical areas do not appear equal when projected into Cartesian coordinates. Consider how the cylindrical projection in the image greatly inflates the area of the poles (e.g., Antarctica). An extensive discussion of solutions to this problem is linked to from the syllabus under **Map Projections**.
It is possible to create mappings in which spherical surfaces are equal-area when projected onto a flat surface.

The solution used most often to display sky areas in astronomy is the Hammer-Aitoff projection (depicted for the Earth in the image).

The Hammer-Aitoff projection is equal area (note how much smaller Antarctica is in this depiction).

More information, including the equations behind the Hammer-Aitoff projection are at the syllabus links.
Map Projections in Python

- Projections such as the Hammer-Aitoff are available in matplotlib

- The general set of commands is
  
  - `import matplotlib.pyplot as plt`
  - `fig = plt.figure()`
  - `ax = fig.add_subplot(111, projection="aitoff")`
    - here 111 means “subplot 1 of a 1x1 grid of plots”
  - `ax.scatter(ra, dec); fig.show()`
    - `ra, dec` here must be in radians with $-\pi < ra < \pi$
Other useful commands and keywords include

- `ax.scatter(ra, dec, marker='o', color='b', s=0.7, alpha=0.5)`
  - here I supplied the points a shape, color, size and opacity
  - the points will be small blue, half-transparent circles

- `xlab = ['14h', '16h', '18h', '20h', '22h', '0h', '2h', '4h', '6h', '8h', '10h']`

- `ax.set_xticklabels(xlab, weight=800)`
  - here I supplied x-axis labels and made them **bold**
  - the point of xlab is to label in **hours** instead of degrees

- `ax.grid(color='k', linestyle='solid', linewidth=0.5)`
  - here I drew a grid of axes of a given style and thickness
  - the grid will be black, solid, and not too thick
1. Generate a random set of 10000 points on the surface of the sphere with coordinates $ra, dec (\alpha, \delta)$ in radians that correctly populate the sphere equally in area

- from numpy.random import random
- $ra = 2*\text{np.pi}*(\text{random}(10000)-0.5)$
- $dec = \text{np.arcsin}(1.-\text{random}(10000)*2.)$
- plot $(ra, dec)$ on a standard $(x, y)$ grid...are there more points near the poles or near the equator of the sphere?

2. Now plot your points in an Aitoff projection

- Change the x-labels to hours instead of degrees
- Add a thick, blue, dashed axis grid using grid
- Change your plot to a Lambert projection
3. Make a plot (binned at 1° in roughly equal areas) to map Galactic dust at $\delta > 0^\circ$ in Aitoff projection

- Generate a grid in $(\alpha, \delta)$ in degrees with $0 < \alpha < 360^\circ$ with larger bins in $\alpha$ (as $1/\cos \delta$) at higher latitude
  - see np.meshgrid from the dust maps lecture
- Determine the values of the reddening (what we called $ebmv$ in the dust maps lecture) at each $(\alpha, \delta)$
- Convert $(\alpha, \delta)$ to $(x, y)$ in Aitoff projection:
  - from astropy import wcs
  - $w = wcs.WCS(naxis=2)$
  - $w.wcs.ctype = ["RA---AIT", "DEC--AIT"]$
  - $x, y = w.wcs_world2pix(ra, dec, 1)$
• Plot \((\alpha, \delta)\) and \((x,y)\)
  
  – *can you see the difference?*

• Use *contour* (see the *dust maps* lecture) to create and plot contours for \(ebmv\) at each \((x,y)\)...these should now be correctly projected

• Plot the Galactic Plane. Does the dust follow the Galaxy?