gcm_analysis

May 14, 2018

1 Guide to analyzing PlaSim output

The first thing we need to do is import a few packages. NumPy is always a must, and matplotlib provides one of the best and most visually-compelling scientific plotting packages out there. PlaSim's postprocessor (burn7) produces netCDF files (.nc), so we import python-netCDF4. Finally, the %matplotlib inline token tells the notebook that we want plots to render within the notebook.

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import netCDF4 as nc
        %matplotlib inline
```

First we'll open the output file, earth.nc. We use the token "r" to tell netCDF4 that we only want to read this file, not modify it. This package also provides support for modifying or writing netCDF4 files, which can be useful, but which we won't go into. This basically opens the file for access, but does not actually load it into memory. That is done when you access the variables stored in the file.

```
In [2]: mydata = nc.Dataset("earth.nc","r")
```

Let's take a look at what's in the file. This will depend on what arguments you pass to the postprocessor, and whether or not you modify PlaSim to output additional variables (some of mine are my own variables).

```
In [3]: print "%-8s %-30s \t%-17s\t%-12s"%("Key","Description","Variable Shape","Units")
        print "%-8s %-30s \t%-17s\t%-12s"%(6*"-",35*"-",19*"-",12*"-")
        for k in mydata.variables:
            ndims = len(mydata.variables[k].shape)
            nspace = 4-ndims
            try:
                print "%-8s %-30s "%(k,mydata.variables[k].long_name)+'\t',\
                      "("+(ndims*"%02d, ")%mydata.variables[k].shape+")"+\
                      (nspace*"%-4s")%(nspace*(" ",)),'\t',mydata.variables[k].units
            except:
                print "%-8s %-30s "%(k,'')+'\t',\
                      "("+(ndims*"%02d, ")%mydata.variables[k].shape+")"+\
                      (nspace*"%-4s")%(nspace*(" ",)),'\t',mydata.variables[k].units
```

Key	Description	Variable Shape	Units
------	------------------------------------	-------------------	------------
lon	longitude	(64,)	degrees_east
lat	latitude	(32,)	degrees_north
lev	sigma at layer midpoints	(10,)	level
time		(12,)	months since 0024-01-
sg	surface_geopotential	(12, 32, 64,)	m2 s-2
ta	air_temperature	(12, 10, 32, 64,)	K
ua	eastward_wind	(12, 10, 32, 64,)	m s-1
va	northward_wind	(12, 10, 32, 64,)	m s-1
hus	specific_humidity	(12, 10, 32, 64,)	1
ps	surface_air_pressure	(12, 32, 64,)	hPa
wap	vertical_air_velocity	(12, 10, 32, 64,)	Pa s-1
wa	upward_wind	(12, 10, 32, 64,)	m s-1
zeta	atm_relative_vorticity	(12, 10, 32, 64,)	s-1
ts	surface_temperature	(12, 32, 64,)	K
mrso	lwe_of_soil_moisture_content	(12, 32, 64,)	m
snd	surface_snow_thickness	(12, 32, 64,)	m
prl	lwe_of_large_scale_precipitatio	(12, 32, 64,)	m s-1
prc	convective_precipitation_rate	(12, 32, 64,)	m s-1
prsn	lwe_of_snowfall_amount	(12, 32, 64,)	m s-1
hfss	surface_sensible_heat_flux	(12, 32, 64,)	W m-2
hfls	surface_latent_heat_flux	(12, 32, 64,)	W m-2
stf	streamfunction	(12, 10, 32, 64,)	m2 s-2
psi	velocity_potential	(12, 10, 32, 64,)	m2 s-2
psl	air_pressure_at_sea_level	(12, 32, 64,)	hPa
pl	log_surface_pressure	(12, 32, 64,)	1
d	divergence_of_wind	(12, 10, 32, 64,)	s-1
hur	relative_humidity	(12, 10, 32, 64,)	1
mrro	surface_runoff	(12, 32, 64,)	m s-1
clw	liquid_water_content	(12, 10, 32, 64,)	1
cl	cloud_area_fraction_in_layer	(12, 10, 32, 64,)	1
clt	cloud_area_fraction	(12, 32, 64,)	1
tas	air_temperature_2m	(12, 32, 64,)	K
tsa	surface_temperature_accumulated	(12, 32, 64,)	K
lsm	land_binary_mask	(12, 32, 64,)	1
z0	surface_roughness_length	(12, 32, 64,)	m
as	surface_albedo	(12, 32, 64,)	1
rss	surface_net_shortwave_flux	(12, 32, 64,)	W m-2
rls	surface_net_longwave_flux	(12, 32, 64,)	W m-2
rst	toa_net_shortwave_flux	(12, 32, 64,)	W m-2
rlut	toa_net_longwave_flux	(12, 32, 64,)	W m-2
evap	lwe_of_water_evaporation	(12, 32, 64,)	m s-1
tso	climate_deep_soil_temperature	(12, 32, 64,)	K
rsut	toa_outgoing_shortwave_flux	(12, 32, 64,)	W m-2
ssru	surface_solar_radiation_upward	(12, 32, 64,)	W m-2
stru	surface_thermal_radiation_upwar	(12, 32, 64,)	W m-2
tso2	soil_temperature_level_2	(12, 32, 64,)	K

```
sic      sea_ice_cover                         (12, 32, 64, )       1
sit      sea_ice_thickness                     (12, 32, 64, )       m
snm      snow_melt                             (12, 32, 64, )       m s-1
sndc     snow_depth_change                     (12, 32, 64, )       m s-1
prw      atmosphere_water_vapor_content        (12, 32, 64, )       kg m-2
glac     glacier_cover                         (12, 32, 64, )       1
spd      wind_speed                            (12, 10, 32, 64, )   m s-1
pr       total_precipitation                   (12, 32, 64, )       m s-1
ntr      net_top_radiation                     (12, 32, 64, )       W m-2
nbr      net_bottom_radiation                  (12, 32, 64, )       W m-2
hfns     surface_downward_heat_flux            (12, 32, 64, )       W m-2
wfn      net_water_flux                        (12, 32, 64, )       m s-1
lwth     local_weathering                      (12, 32, 64, )       W_earth
grnz     ground_geopotential                   (12, 32, 64, )       m2 s-2
icez     glacier_geopotential                  (12, 32, 64, )       m2 s-2
netz     net_geopotential                      (12, 32, 64, )       m2 s-2
czen     cosine_solar_zenith_angle             (12, 32, 64, )       nondimen
wthpr    weatherable_precipitation             (12, 32, 64, )       mm day-1
mint     minimum_temperature                   (12, 32, 64, )       K
maxt     maximum_temperature                   (12, 32, 64, )       K
```

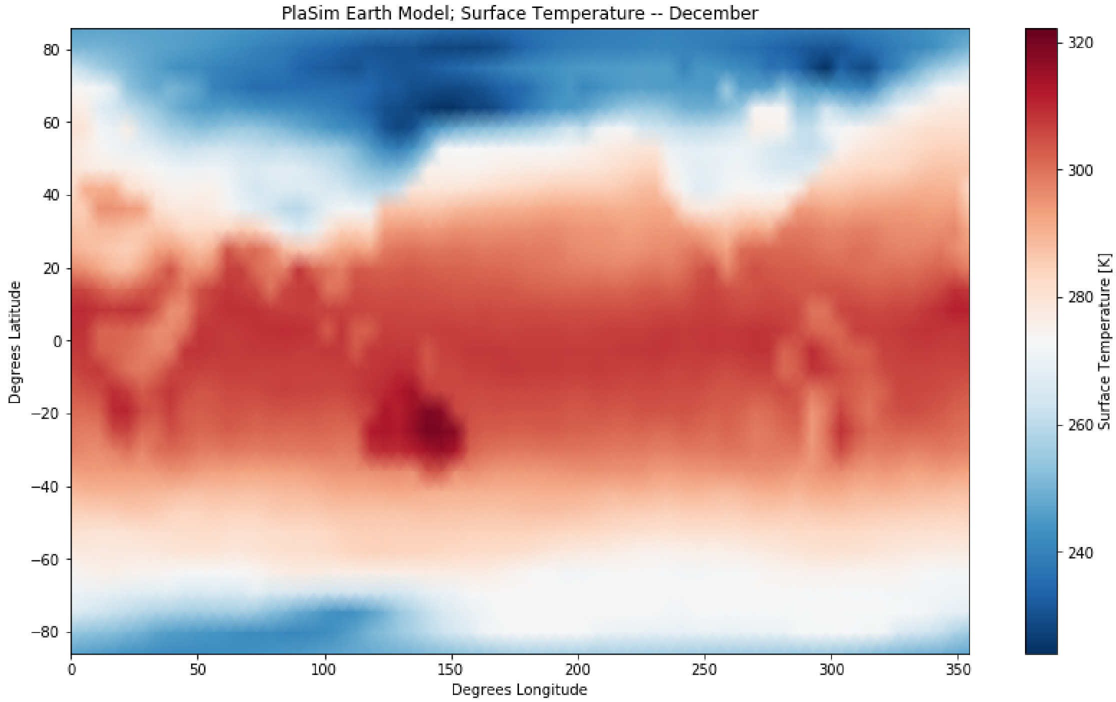
1.1 Plotting Data

If we want to plot some data, we'll first have to get the latitude and longitude data. If we only want the data as a NumPy array, we need to 'slice' the variable--otherwise we'll just be getting a netCDF4 variable object.

```
In [4]: lat = mydata.variables['lat'][:]
        lon = mydata.variables['lon'][:]
```

Notice that each variable has a length of 12 in its first dimension--there are 12 months in this file, and we have monthly averages for each variable. They run January-December: a calendar year. Let's take a look at the average surface temperature in December. I'll be using colormaps from https://matplotlib.org/users/colormaps.html.

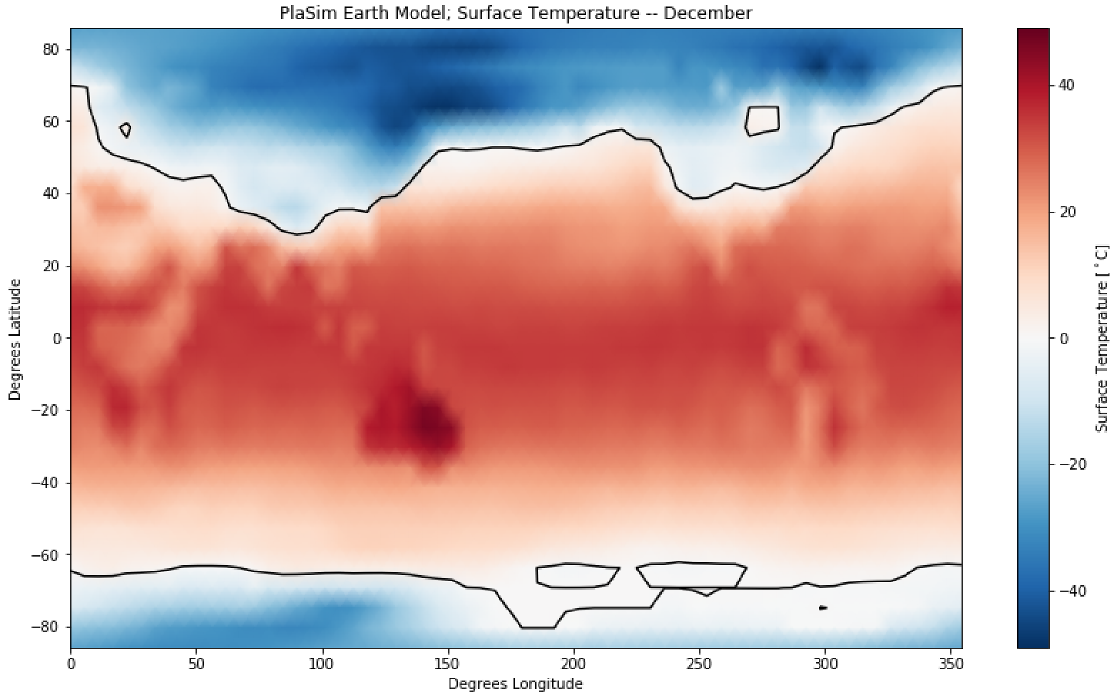
```
In [5]: fig,ax = plt.subplots(figsize=(14,8))
        x=mydata.variables['ts'][-1,:,:] #Let's only plot the most recent month
        tmin = 273.15 - np.amax(abs(x-273.15)) #This ensures that our data range is
        tmax = 273.15 + np.amax(abs(x-273.15)) #centered on the freezing point, but
                                               #includes all temperatures in the model.
        im=plt.pcolormesh(lon,lat,x,cmap='RdBu_r',shading='Gouraud',vmin=tmin,vmax=tmax)
        #The shading argument means we provide coordinates for cell centers, do a bit of interpo
        plt.colorbar(im,label="Surface Temperature [K]")
        plt.xlabel("Degrees Longitude")
        plt.ylabel("Degrees Latitude")
        plt.title("PlaSim Earth Model; Surface Temperature -- December")

Out[5]: Text(0.5,1,u'PlaSim Earth Model; Surface Temperature -- December')
```



Let's make it Celsius, and add a contour indicating the 0-degree isotherm:

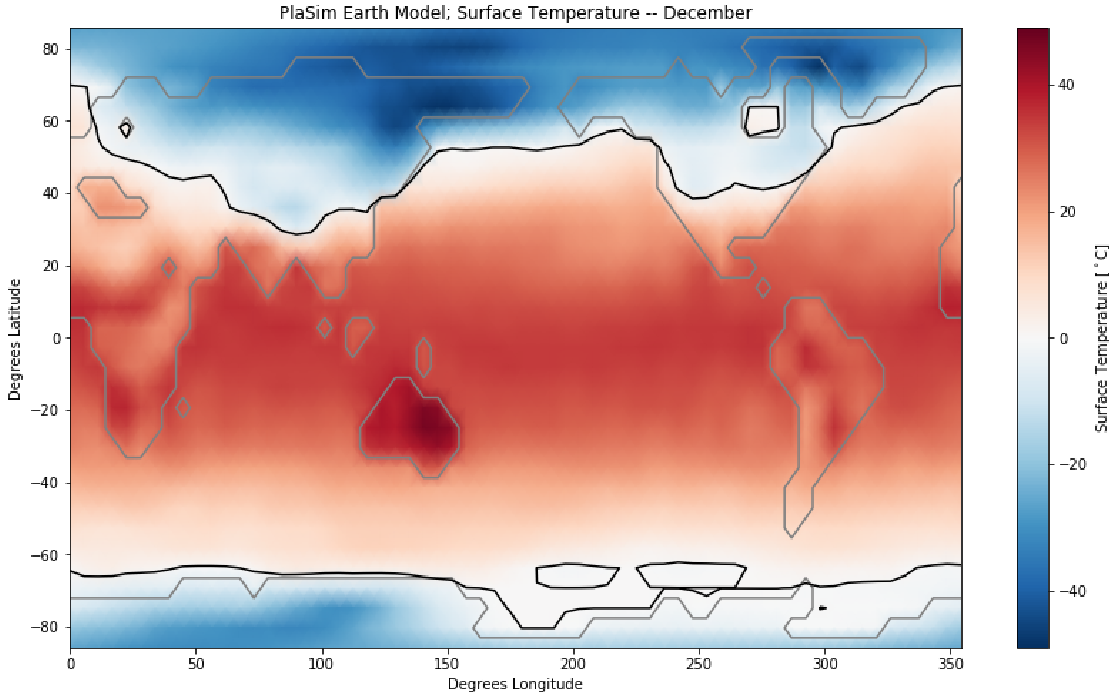
In [6]:
```
fig,ax = plt.subplots(figsize=(14,8))
x=mydata.variables['ts'][-1,:,:] #Let's only plot the most recent month
tmin = -np.amax(abs(x-273.15))
tmax = np.amax(abs(x-273.15))
im=plt.pcolormesh(lon,lat,x-273.15,cmap='RdBu_r',shading='Gouraud',vmin=tmin,vmax=tmax)
plt.contour(lon,lat,x-273.15,(0,),colors='k') #Black zero-degree isotherm
plt.colorbar(im,label="Surface Temperature [$^\circ$C]") #Notice some support for LaTeX
plt.xlabel("Degrees Longitude")
plt.ylabel("Degrees Latitude")
plt.title("PlaSim Earth Model; Surface Temperature -- December")
```

Out[6]:
```
Text(0.5,1,u'PlaSim Earth Model; Surface Temperature -- December')
```



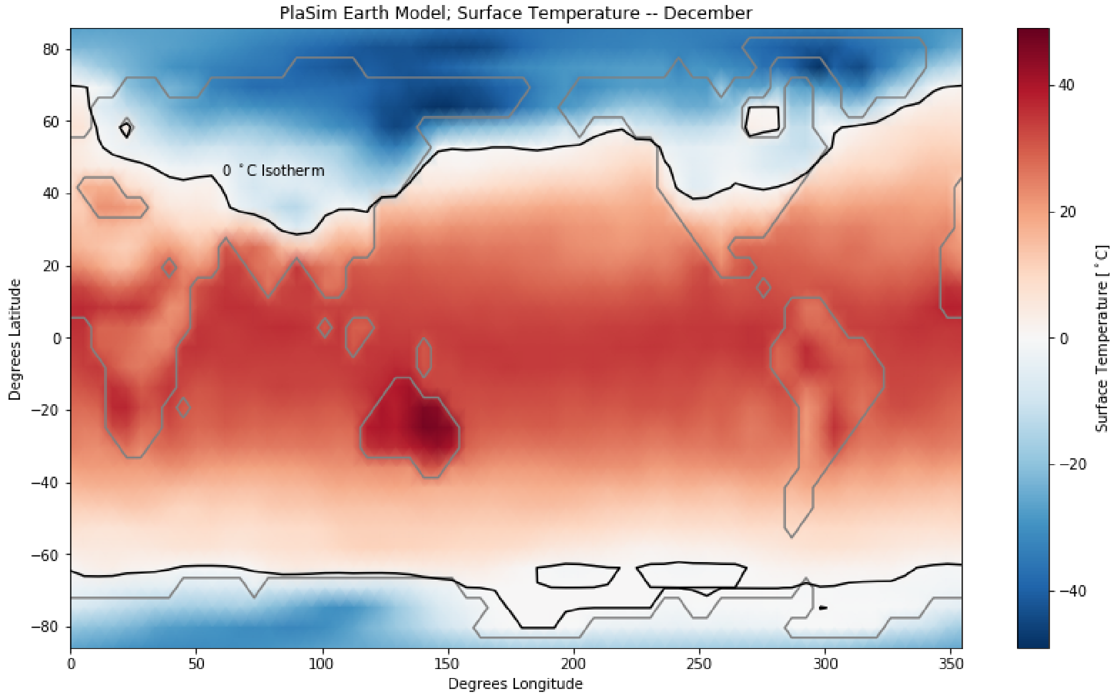
We can kind of see where the continents are, but let's make it clearer:

```
In [7]: fig,ax = plt.subplots(figsize=(14,8))
        x=mydata.variables['ts'][-1,:,:] #Let's only plot the most recent month
        tmin = -np.amax(abs(x-273.15))
        tmax = np.amax(abs(x-273.15))
        im=plt.pcolormesh(lon,lat,x-273.15,cmap='RdBu_r',shading='Gouraud',vmin=tmin,vmax=tmax)

        lsm = mydata.variables['lsm'][-1,:,:]
        plt.contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries

        plt.contour(lon,lat,x-273.15,(0,),colors='k') #Black zero-degree isotherm
        plt.colorbar(im,label="Surface Temperature [$^\circ$C]")
        plt.xlabel("Degrees Longitude")
        plt.ylabel("Degrees Latitude")
        plt.title("PlaSim Earth Model; Surface Temperature -- December")

Out[7]: Text(0.5,1,u'PlaSim Earth Model; Surface Temperature -- December')
```



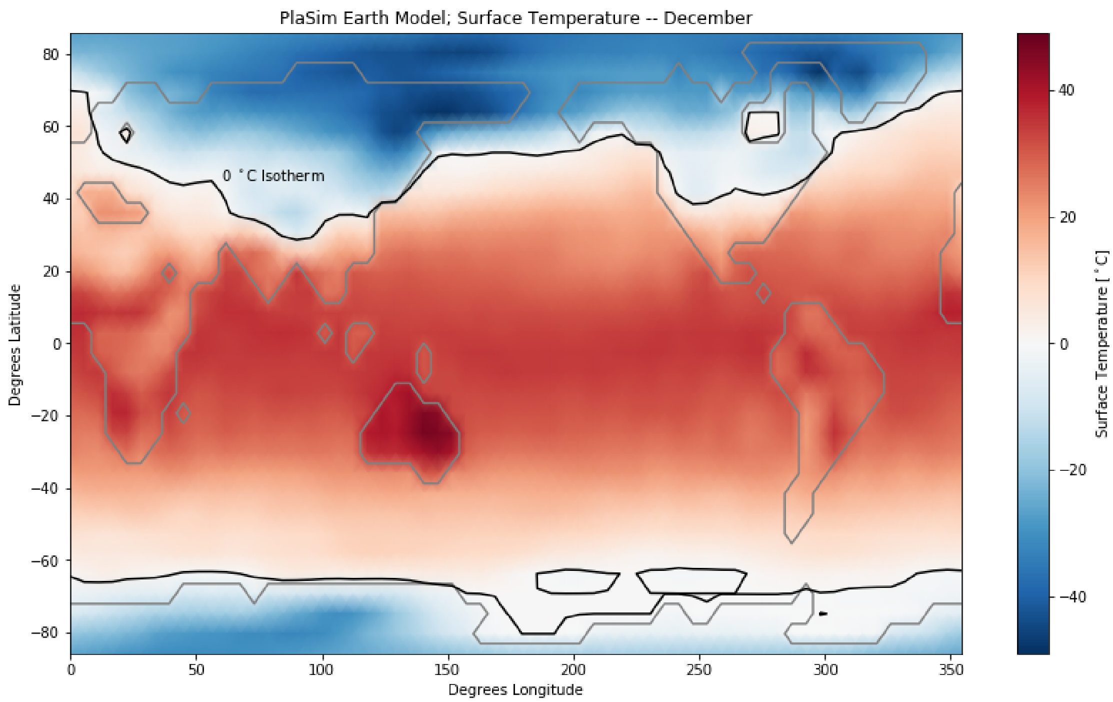
Maybe we should add a label to the isotherm....

```
In [8]: fig,ax = plt.subplots(figsize=(14,8))
        x=mydata.variables['ts'][-1,:,:]
        tmin = -np.amax(abs(x-273.15))
        tmax = np.amax(abs(x-273.15))
        im=plt.pcolormesh(lon,lat,x-273.15,cmap='RdBu_r',shading='Gouraud',vmin=tmin,vmax=tmax)
        plt.contour(lon,lat,lsm,(0.5,),colors='gray')
        plt.contour(lon,lat,x-273.15,(0,),colors='k')
        plt.colorbar(im,label="Surface Temperature [$^\circ$C]")
        plt.xlabel("Degrees Longitude")
        plt.ylabel("Degrees Latitude")
        plt.title("PlaSim Earth Model; Surface Temperature -- December")
        plt.annotate("0 $^\circ$C Isotherm",xy=(60,45),xytext=(60,45)) #Notice coords are in dat

Out[8]: Text(60,45,u'0 $^\\circ$C Isotherm')
```



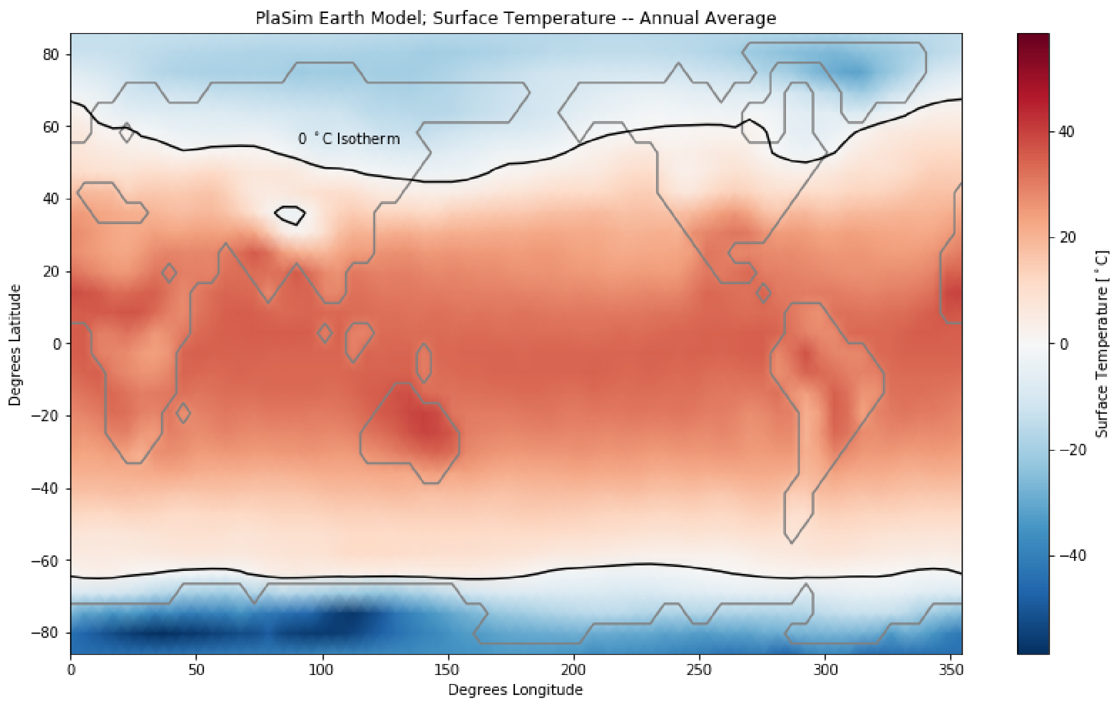
That looks pretty good, so let's save it.

```
In [9]: fig,ax = plt.subplots(figsize=(14,8))
        x=mydata.variables['ts'][-1,:,:]
        tmin = -np.amax(abs(x-273.15))
        tmax = np.amax(abs(x-273.15))
        im=plt.pcolormesh(lon,lat,x-273.15,cmap='RdBu_r',shading='Gouraud',vmin=tmin,vmax=tmax)
        plt.contour(lon,lat,lsm,(0.5,),colors='gray')
        plt.contour(lon,lat,x-273.15,(0,),colors='k')
        plt.colorbar(im,label="Surface Temperature [$^\circ$C]")
        plt.xlabel("Degrees Longitude")
        plt.ylabel("Degrees Latitude")
        plt.title("PlaSim Earth Model; Surface Temperature -- December")
        plt.annotate("0 $^\circ$C Isotherm",xy=(60,45),xytext=(60,45))
        plt.savefig("mytemp_plot.pdf",bbox_inches='tight') #The bbox_inches arg reduces white ma
```



We can also look at the annual average temperature:

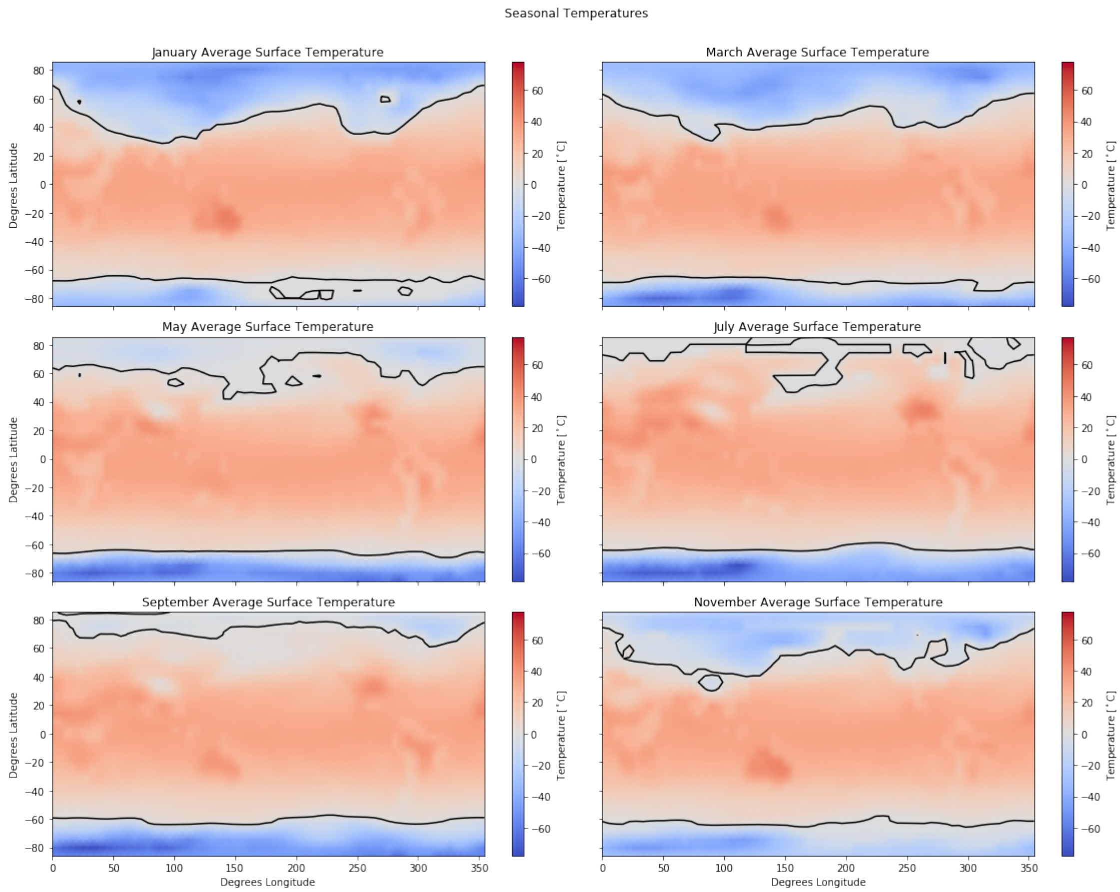
```
In [10]: fig,ax = plt.subplots(figsize=(14,8))

         x=np.mean(mydata.variables['ts'][:,:,:],axis=0) #Average over the first axis, time.

         tmin = -np.amax(abs(x-273.15)) #
         tmax = np.amax(abs(x-273.15))
         im=plt.pcolormesh(lon,lat,x-273.15,cmap='RdBu_r',shading='Gouraud',vmin=tmin,vmax=tmax)
         plt.contour(lon,lat,lsm,(0.5,),colors='gray')
         plt.contour(lon,lat,x-273.15,(0,),colors='k')
         plt.colorbar(im,label="Surface Temperature [$^\circ$C]")
         plt.xlabel("Degrees Longitude")
         plt.ylabel("Degrees Latitude")
         plt.title("PlaSim Earth Model; Surface Temperature -- Annual Average")
         plt.annotate("0 $^\circ$C Isotherm",xy=(90,55),xytext=(90,55))

Out[10]: Text(90,55,u'0 $^\\circ$C Isotherm')
```



Let's look at some seasonal snapshots:

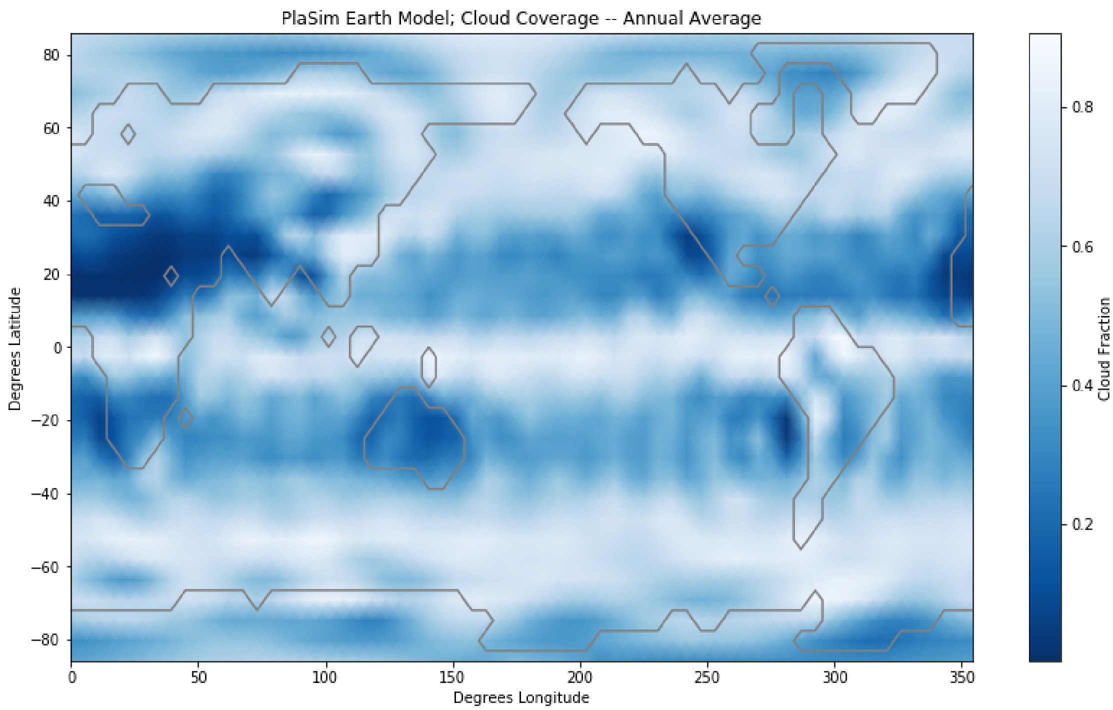
```
In [11]: fig,axes = plt.subplots(3,2,figsize=(16,13),sharex=True,sharey=True)

         x = mydata.variables['ts'][:,:,:] - 273.15

         tmin = -np.amax(abs(x))
         tmax = np.amax(abs(x))

         im=axes[0,0].pcolormesh(lon,lat,x[0],cmap='coolwarm',shading='Gouraud',
                                 vmin=tmin,vmax=tmax)
         axes[0,0].contour(lon,lat,x[0],(0,),colors='k')
         axes[0,0].set_title("January Average Surface Temperature")
         axes[0,0].set_ylabel("Degrees Latitude")
         axes[0,1].pcolormesh(lon,lat,x[2],cmap='coolwarm',shading='Gouraud',
                              vmin=tmin,vmax=tmax)
         axes[0,1].contour(lon,lat,x[2],(0,),colors='k')
         axes[0,1].set_title("March Average Surface Temperature")
         plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[0,0])
         plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[0,1])

         axes[1,0].pcolormesh(lon,lat,x[4],cmap='coolwarm',shading='Gouraud',
                              vmin=tmin,vmax=tmax)
         axes[1,0].contour(lon,lat,x[4],(0,),colors='k')
         axes[1,0].set_title("May Average Surface Temperature")
         axes[1,0].set_ylabel("Degrees Latitude")
```

```
axes[1,1].pcolormesh(lon,lat,x[6],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[1,1].contour(lon,lat,x[6],(0,),colors='k')
axes[1,1].set_title("July Average Surface Temperature")
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[1,0])
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[1,1])

axes[2,0].pcolormesh(lon,lat,x[8],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[2,0].contour(lon,lat,x[8],(0,),colors='k')
axes[2,0].set_title("September Average Surface Temperature")
axes[2,0].set_ylabel("Degrees Latitude")
axes[2,0].set_xlabel("Degrees Longitude")
axes[2,1].pcolormesh(lon,lat,x[10],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[2,1].contour(lon,lat,x[10],(0,),colors='k')
axes[2,1].set_title("November Average Surface Temperature")
axes[2,1].set_xlabel("Degrees Longitude")
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[2,0])
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[2,1])

fig.suptitle("Seasonal Temperatures")
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```



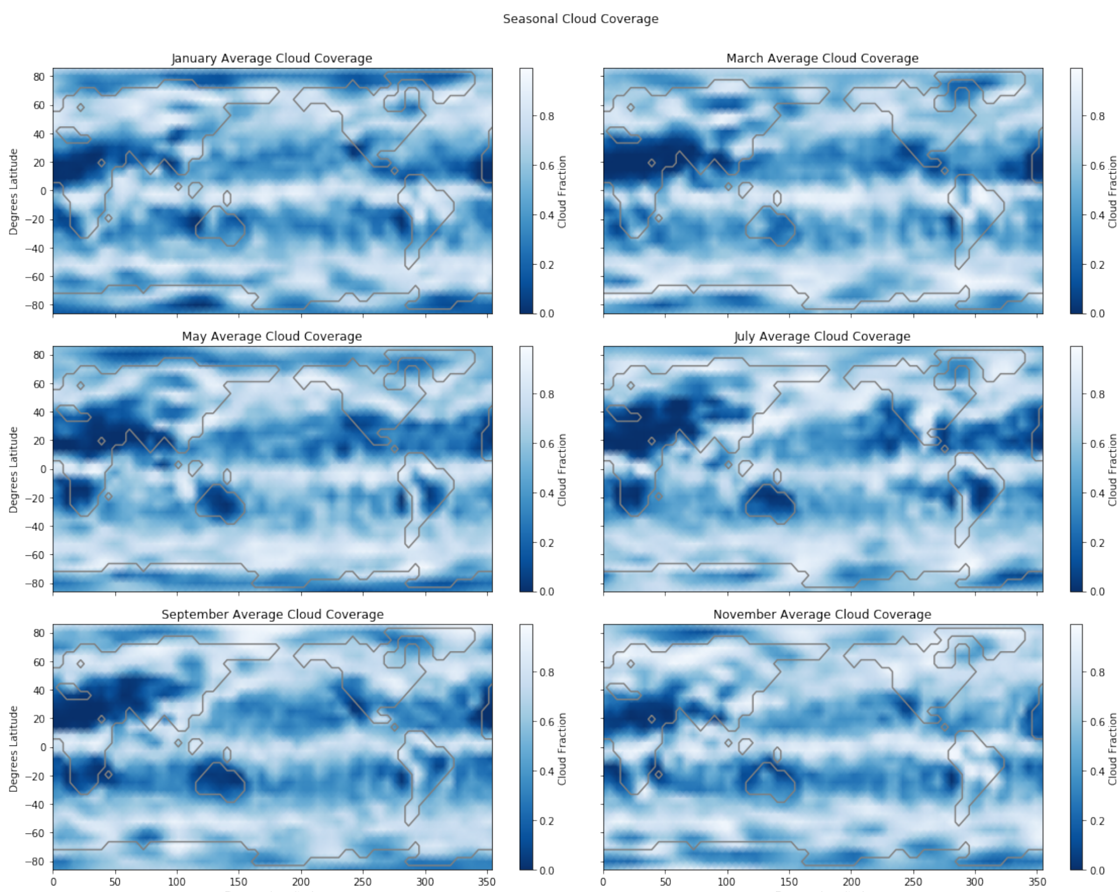
Now let's look at some other variables. How about clouds?

```
In [12]: fig,ax = plt.subplots(figsize=(14,8))

         x=np.mean(mydata.variables['clt'][:,:,:],axis=0)

         im=plt.pcolormesh(lon,lat,x,cmap='Blues_r',shading='Gouraud')
         plt.contour(lon,lat,lsm,(0.5,),colors='gray')
         plt.colorbar(im,label="Cloud Fraction")
         plt.xlabel("Degrees Longitude")
         plt.ylabel("Degrees Latitude")
         plt.title("PlaSim Earth Model; Cloud Coverage -- Annual Average")

Out[12]: Text(0.5,1,u'PlaSim Earth Model; Cloud Coverage -- Annual Average')
```




```
In [13]: fig,axes = plt.subplots(3,2,figsize=(16,13),sharex=True,sharey=True)

         x = mydata.variables['clt'][:,:,:]


         im=axes[0,0].pcolormesh(lon,lat,x[0],cmap='Blues_r',shading='Gouraud')
         axes[0,0].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
         axes[0,0].set_title("January Average Cloud Coverage")
         axes[0,0].set_ylabel("Degrees Latitude")
         axes[0,1].pcolormesh(lon,lat,x[2],cmap='Blues_r',shading='Gouraud')
         axes[0,1].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
         axes[0,1].set_title("March Average Cloud Coverage")
         plt.colorbar(im,label="Cloud Fraction",ax=axes[0,0])
         plt.colorbar(im,label="Cloud Fraction",ax=axes[0,1])

         axes[1,0].pcolormesh(lon,lat,x[4],cmap='Blues_r',shading='Gouraud')
         axes[1,0].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
         axes[1,0].set_title("May Average Cloud Coverage")
         axes[1,0].set_ylabel("Degrees Latitude")
         axes[1,1].pcolormesh(lon,lat,x[6],cmap='Blues_r',shading='Gouraud')
         axes[1,1].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
         axes[1,1].set_title("July Average Cloud Coverage")
         plt.colorbar(im,label="Cloud Fraction",ax=axes[1,0])
         plt.colorbar(im,label="Cloud Fraction",ax=axes[1,1])
```

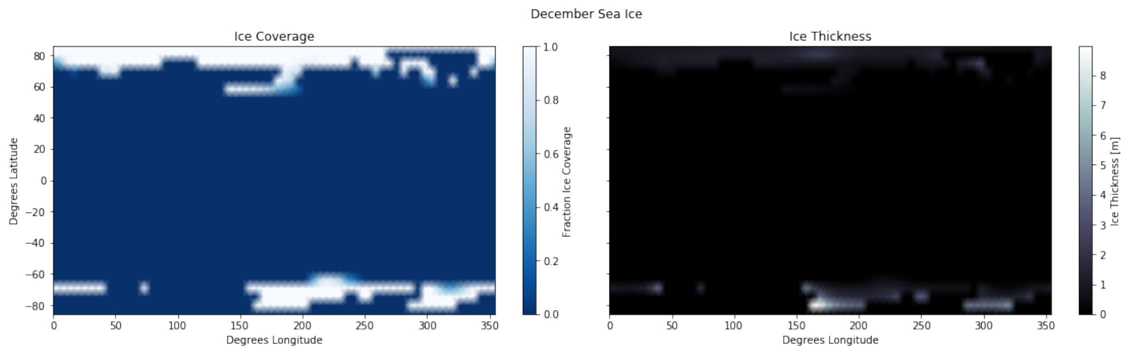
```
axes[2,0].pcolormesh(lon,lat,x[8],cmap='Blues_r',shading='Gouraud')
axes[2,0].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
axes[2,0].set_title("September Average Cloud Coverage")
axes[2,0].set_ylabel("Degrees Latitude")
axes[2,0].set_xlabel("Degrees Longitude")
axes[2,1].pcolormesh(lon,lat,x[10],cmap='Blues_r',shading='Gouraud')
axes[2,1].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
axes[2,1].set_title("November Average Cloud Coverage")
axes[2,1].set_xlabel("Degrees Longitude")
plt.colorbar(im,label="Cloud Fraction",ax=axes[2,0])
plt.colorbar(im,label="Cloud Fraction",ax=axes[2,1])

fig.suptitle("Seasonal Cloud Coverage")
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```



Let's take a look at December sea ice coverage and thickness.

```
In [14]: fig,axes=plt.subplots(1,2,figsize=(16,5),sharey=True,squeeze=True)
```

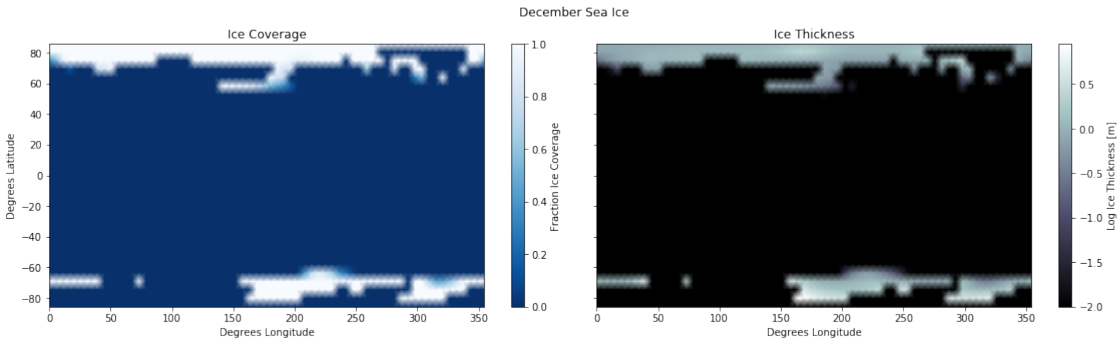
```
x = mydata.variables['sic'][-1,:,:]
y = mydata.variables['sit'][-1,:,:]

fig.suptitle("December Sea Ice")
axes[0].set_title("Ice Coverage")
axes[1].set_title("Ice Thickness")
axes[0].set_xlabel("Degrees Longitude")
axes[0].set_ylabel("Degrees Latitude")
axes[1].set_xlabel("Degrees Longitude")

im1 = axes[0].pcolormesh(lon,lat,x,cmap='Blues_r',shading='Gouraud')
plt.colorbar(im1,label="Fraction Ice Coverage",ax=axes[0])
im2 = axes[1].pcolormesh(lon,lat,y,cmap='bone',shading='Gouraud')
plt.colorbar(im2,label="Ice Thickness [m]",ax=axes[1])

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```



That ice thickness is kind of hard to see--maybe it's thin ice? Let's check by changing the scaling.

```
In [15]: fig,axes=plt.subplots(1,2,figsize=(16,5),sharey=True,squeeze=True)

         x = mydata.variables['sic'][-1,:,:]
         y = np.log10(np.maximum(mydata.variables['sit'][-1,:,:],1.0e-2))
         #Log(0) is undefined, so we define a minimum--we don't care about
         #ice thinner than 1cm.

         fig.suptitle("December Sea Ice")
         axes[0].set_title("Ice Coverage")
         axes[1].set_title("Ice Thickness")
         axes[0].set_xlabel("Degrees Longitude")
         axes[0].set_ylabel("Degrees Latitude")
         axes[1].set_xlabel("Degrees Longitude")

         im1 = axes[0].pcolormesh(lon,lat,x,cmap='Blues_r',shading='Gouraud')
```

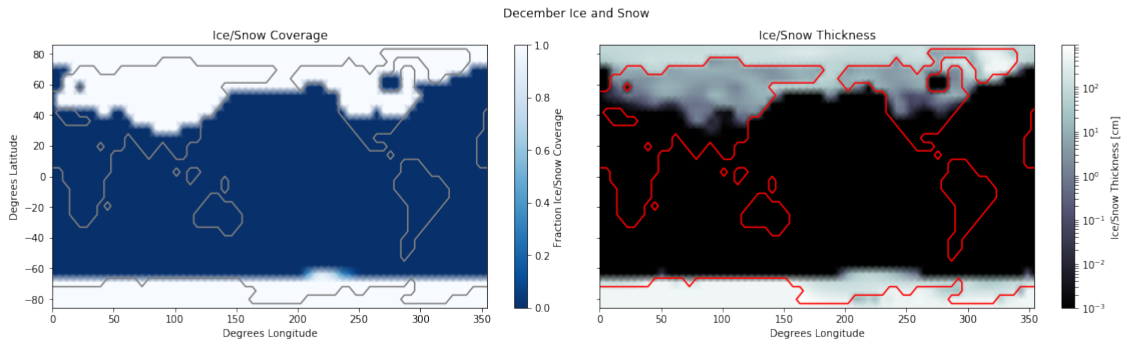
```
plt.colorbar(im1,label="Fraction Ice Coverage",ax=axes[0])
im2 = axes[1].pcolormesh(lon,lat,y,cmap='bone',shading='Gouraud')
plt.colorbar(im2,label="Log Ice Thickness [m]",ax=axes[1])

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```



Let's try combining variables and look at *snow* coverage as well.

```
In [16]: import matplotlib.colors as colors
```

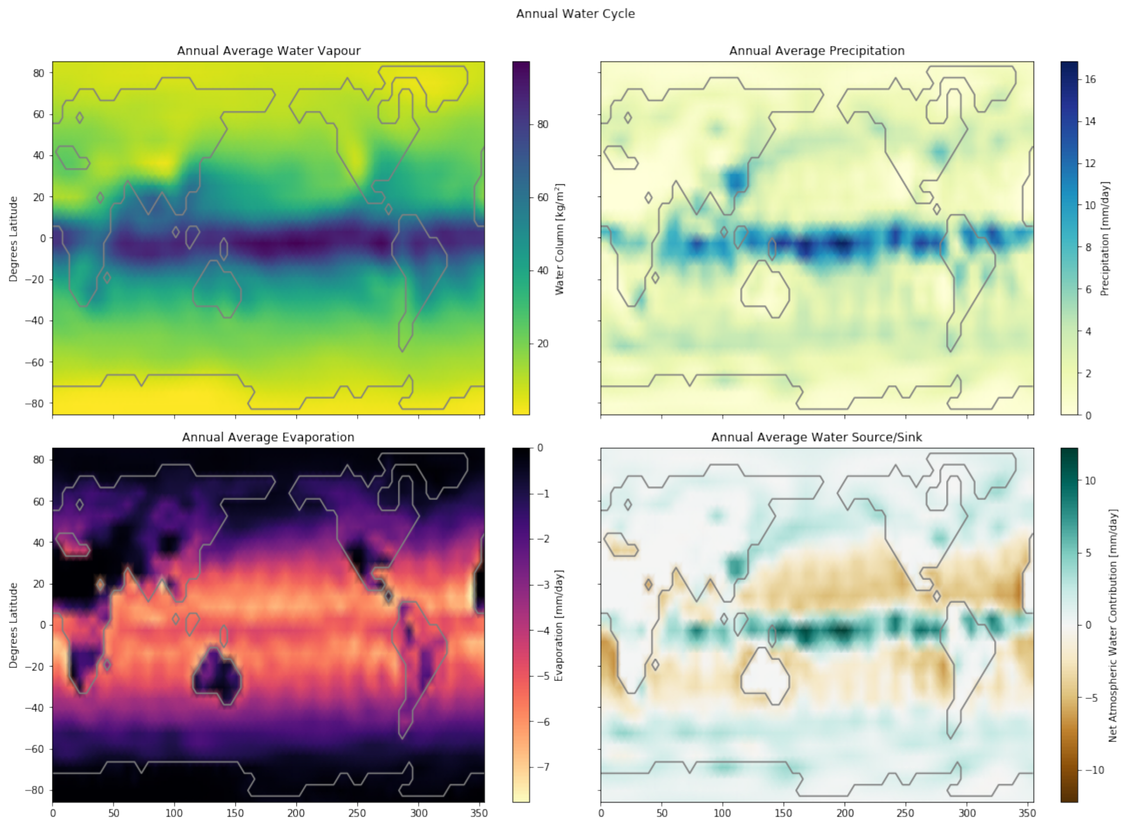
```
In [17]: fig,axes=plt.subplots(1,2,figsize=(16,5),sharey=True,squeeze=True)

         x = np.minimum(mydata.variables['sic'][-1,:,:] +
                        1.0*(mydata.variables['snd'][-1,:,:]>0), 1.0)
         #Limit to 1
         y = np.maximum(mydata.variables['sit'][-1,:,:] +
                        mydata.variables['snd'][-1,:,:], 1.0e-5)*100
         #We don't care about less than 10 microns of snow.

         fig.suptitle("December Ice and Snow")
         axes[0].set_title("Ice/Snow Coverage")
         axes[1].set_title("Ice/Snow Thickness")
         axes[0].set_xlabel("Degrees Longitude")
         axes[0].set_ylabel("Degrees Latitude")
         axes[1].set_xlabel("Degrees Longitude")

         im1 = axes[0].pcolormesh(lon,lat,x,cmap='Blues_r',shading='Gouraud')
         axes[0].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
         plt.colorbar(im1,label="Fraction Ice/Snow Coverage",ax=axes[0])
         im2 = axes[1].pcolormesh(lon,lat,y,cmap='bone',shading='Gouraud',
                                  norm=colors.LogNorm(vmin=y.min(), vmax=y.max()))
         axes[1].contour(lon,lat,lsm,(0.5,),colors='red') #Continent boundaries
         plt.colorbar(im2,label="Ice/Snow Thickness [cm]",ax=axes[1])

         plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```



Sometimes it might be useful to do analysis by comparing humidity, precipitation, evaporation, and net sources and sinks of atmospheric water.

In [18]:
```
fig,axes = plt.subplots(2,2,figsize=(16,12),sharex=True,sharey=True)

w = np.mean(mydata.variables['prw'][:,:,:],axis=0) #Total water content (kg/m^2)
x = np.mean(mydata.variables['pr'][:,:,:],axis=0) * 8.64e7
#precipitation, scaled from m/s to mm/day
y = np.mean(mydata.variables['evap'][:,:,:],axis=0) * 8.64e7
#evaporation, scaled from m/s to mm/day
z = x+y #precipitation minus evaporation
zmin = -np.amax(abs(z))
zmax =  np.amax(abs(z))

im1=axes[0,0].pcolormesh(lon,lat,w,cmap='viridis_r',shading='Gouraud')
axes[0,0].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
axes[0,0].set_title("Annual Average Water Vapour")
axes[0,0].set_ylabel("Degrees Latitude")
plt.colorbar(im1,label="Water Column [kg/m$^2$]",ax=axes[0,0])

im2=axes[0,1].pcolormesh(lon,lat,x,cmap='YlGnBu',shading='Gouraud')
axes[0,1].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
axes[0,1].set_title("Annual Average Precipitation")
plt.colorbar(im2,label="Precipitation [mm/day]",ax=axes[0,1])

im3=axes[1,0].pcolormesh(lon,lat,y,cmap='magma_r',shading='Gouraud')
axes[1,0].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
axes[1,0].set_title("Annual Average Evaporation")
axes[1,0].set_ylabel("Degrees Latitude")
plt.colorbar(im3,label="Evaporation [mm/day]",ax=axes[1,0])

im4=axes[1,1].pcolormesh(lon,lat,z,cmap='BrBG',shading='Gouraud',vmin=zmin,vmax=zmax)
axes[1,1].contour(lon,lat,lsm,(0.5,),colors='gray') #Continent boundaries
axes[1,1].set_title("Annual Average Water Source/Sink")
plt.colorbar(im4,label="Net Atmospheric Water Contribution [mm/day]",ax=axes[1,1])
```

```
fig.suptitle("Annual Water Cycle")
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

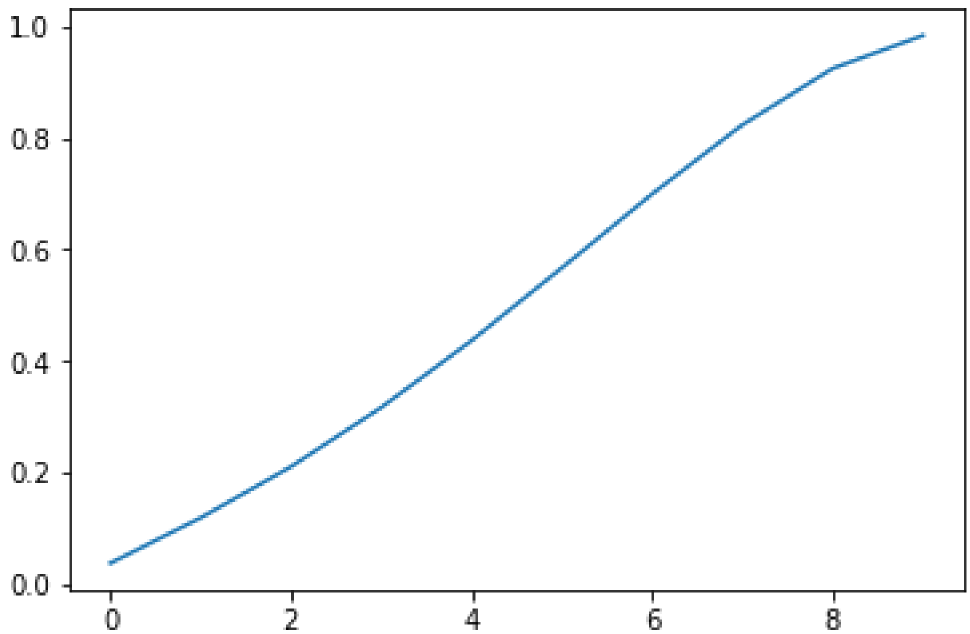


1.1.1 Data with Multiple Atmospheric Levels

Some variables have a vertical dimension, so you know how it varies throughout the atmosphere. Let's take a look at different latitude-longitude slices of air temperature and total wind speed.

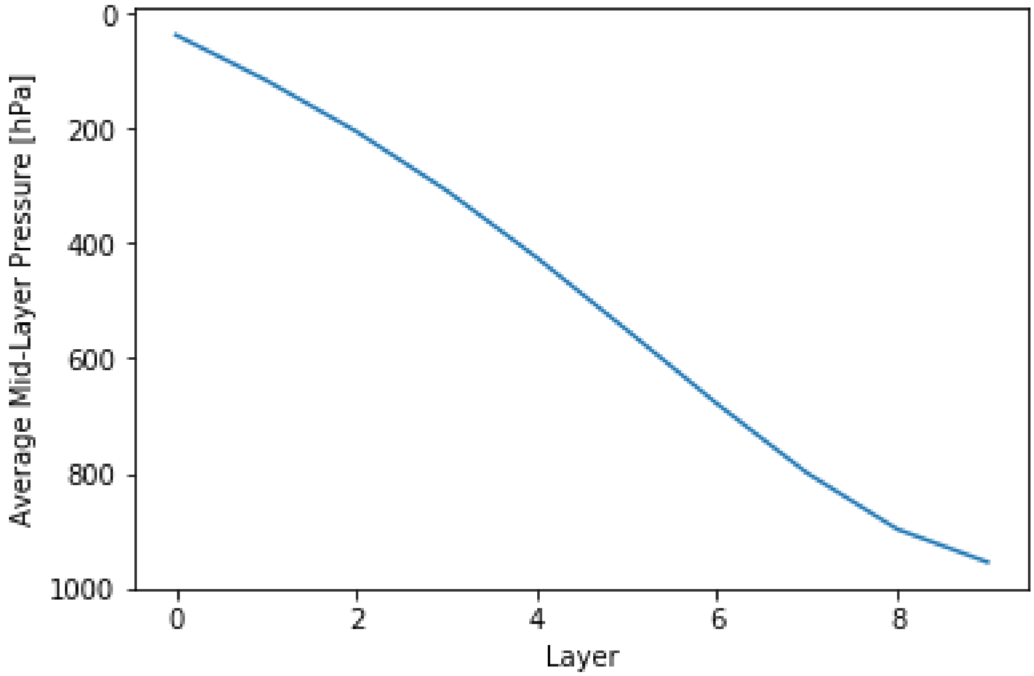
First, we need to understand how the level dimension works. PlaSim uses a sigma-model, which is defined as $\sigma_n = \frac{p_n}{p_s}$, where p_s is the surface pressure, so that $\sigma = 1$ corresponds to the surface.

```
In [19]: lev = mydata.variables['lev'][:]

In [20]: plt.plot(lev)

Out[20]: [<matplotlib.lines.Line2D at 0x7fd55dd2a950>]
```



So the last indices in the vertical dimension are closest to the surface. We can use this to get the average pressure of each layer (in a model with topography, this is useless: the atmospheric layers *follow the topography* (that's the point of the sigma model), so pressure changes a lot within a layer).

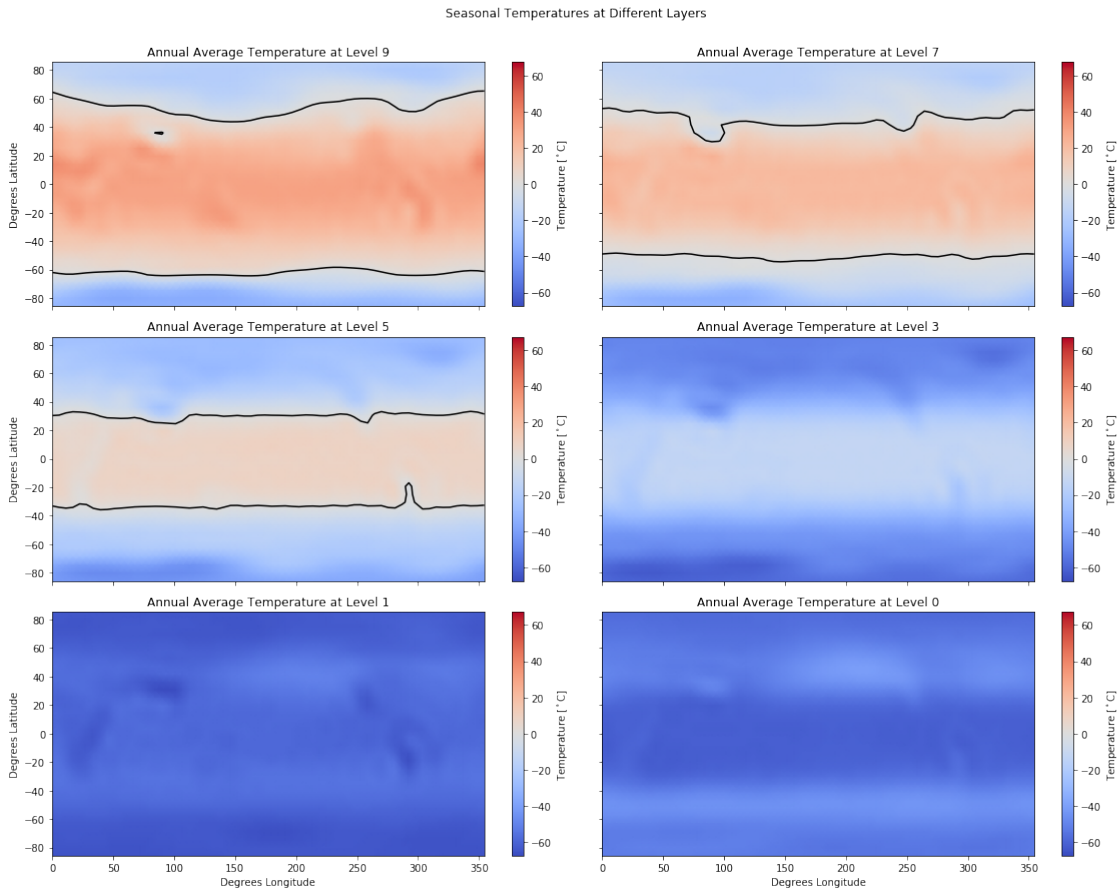
```
In [21]: plt.plot(lev*np.mean(mydata.variables['ps'][:]))
         plt.ylabel("Average Mid-Layer Pressure [hPa]")
         plt.gca().invert_yaxis()
         plt.xlabel("Layer")

Out[21]: Text(0.5,0,u'Layer')
```



In [22]:
```
fig,axes = plt.subplots(3,2,figsize=(16,13),sharex=True,sharey=True)

x = np.mean(mydata.variables['ta'][:,:,:,:],axis=0) - 273.15 #Annual average

tmin = -np.amax(abs(x))
tmax = np.amax(abs(x))

im=axes[0,0].pcolormesh(lon,lat,x[9],cmap='coolwarm',shading='Gouraud',
                        vmin=tmin,vmax=tmax)
axes[0,0].contour(lon,lat,x[9],(0,),colors='k')
axes[0,0].set_title("Annual Average Temperature at Level 9")
axes[0,0].set_ylabel("Degrees Latitude")
axes[0,1].pcolormesh(lon,lat,x[7],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[0,1].contour(lon,lat,x[7],(0,),colors='k')
axes[0,1].set_title("Annual Average Temperature at Level 7")
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[0,0])
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[0,1])

axes[1,0].pcolormesh(lon,lat,x[5],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[1,0].contour(lon,lat,x[5],(0,),colors='k')
axes[1,0].set_title("Annual Average Temperature at Level 5")
axes[1,0].set_ylabel("Degrees Latitude")
```

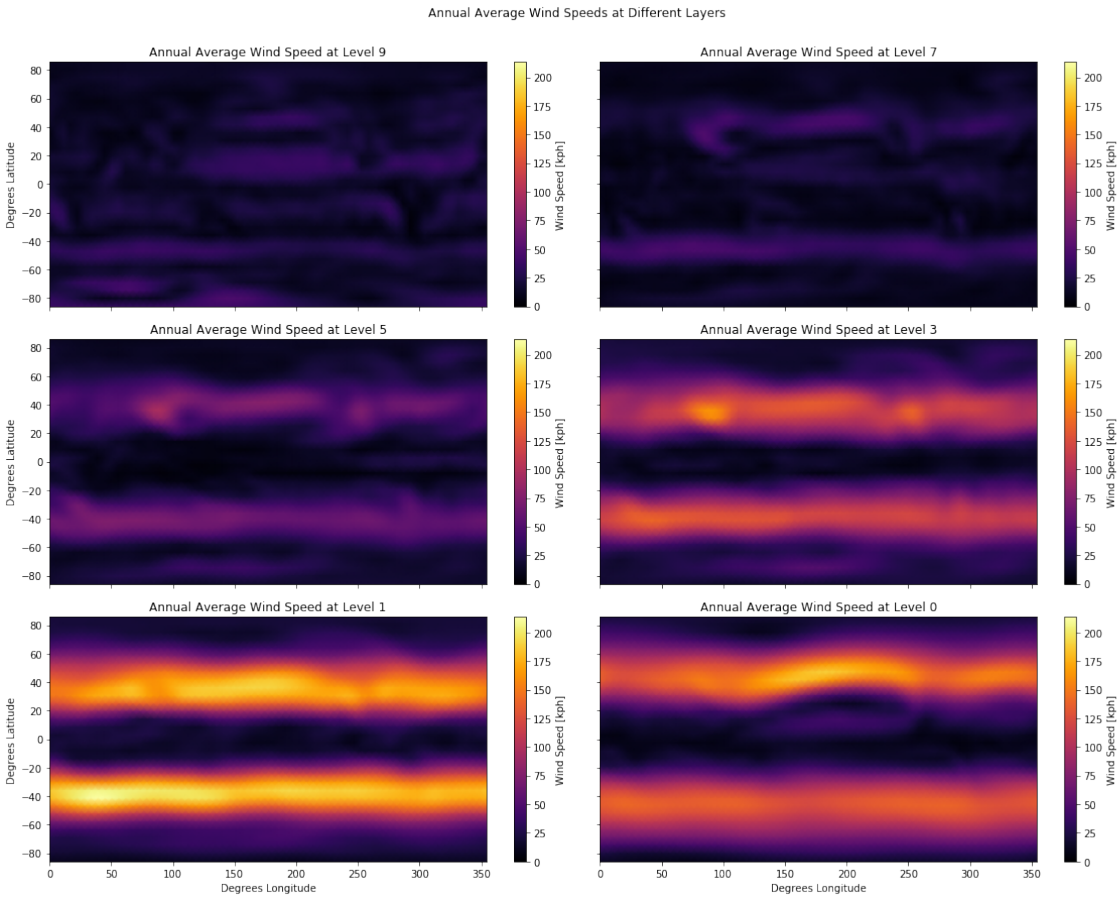
```python
axes[1,1].pcolormesh(lon,lat,x[3],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[1,1].contour(lon,lat,x[3],(0,),colors='k')
axes[1,1].set_title("Annual Average Temperature at Level 3")
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[1,0])
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[1,1])

axes[2,0].pcolormesh(lon,lat,x[1],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[2,0].contour(lon,lat,x[1],(0,),colors='k')
axes[2,0].set_title("Annual Average Temperature at Level 1")
axes[2,0].set_ylabel("Degrees Latitude")
axes[2,0].set_xlabel("Degrees Longitude")
axes[2,1].pcolormesh(lon,lat,x[0],cmap='coolwarm',shading='Gouraud',
                     vmin=tmin,vmax=tmax)
axes[2,1].contour(lon,lat,x[0],(0,),colors='k')
axes[2,1].set_title("Annual Average Temperature at Level 0")
axes[2,1].set_xlabel("Degrees Longitude")
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[2,0])
plt.colorbar(im,label="Temperature [$^\circ$C]",ax=axes[2,1])

fig.suptitle("Seasonal Temperatures at Different Layers")
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

```
/usr/local/lib/python2.7/dist-packages/matplotlib/contour.py:1180: UserWarning: No contour level
  warnings.warn("No contour levels were found"
```




```
In [23]: fig,axes = plt.subplots(3,2,figsize=(16,13),sharex=True,sharey=True)

         us = mydata.variables["ua"][:] #Eastward wind [m/s]
         vs = mydata.variables['va'][:] #Northward wind [m/s]
         ws = mydata.variables["wa"][:] #Upward wind [m/s]

         spd = np.mean(np.sqrt(us**2 + vs**2 + ws**2),axis=0) * 3.6 #Annual Average in kph

         smin = 0
         smax = np.amax(spd)

         im=axes[0,0].pcolormesh(lon,lat,spd[9],cmap='inferno',shading='Gouraud',
                                 vmin=smin,vmax=smax)
         axes[0,0].set_title("Annual Average Wind Speed at Level 9")
         axes[0,0].set_ylabel("Degrees Latitude")
         axes[0,1].pcolormesh(lon,lat,spd[7],cmap='inferno',shading='Gouraud',
                              vmin=smin,vmax=smax)
         axes[0,1].set_title("Annual Average Wind Speed at Level 7")
         plt.colorbar(im,label="Wind Speed [kph]",ax=axes[0,0])
```

```
plt.colorbar(im,label="Wind Speed [kph]",ax=axes[0,1])

axes[1,0].pcolormesh(lon,lat,spd[5],cmap='inferno',shading='Gouraud',
                     vmin=smin,vmax=smax)
axes[1,0].set_title("Annual Average Wind Speed at Level 5")
axes[1,0].set_ylabel("Degrees Latitude")
axes[1,1].pcolormesh(lon,lat,spd[3],cmap='inferno',shading='Gouraud',
                     vmin=smin,vmax=smax)
axes[1,1].set_title("Annual Average Wind Speed at Level 3")
plt.colorbar(im,label="Wind Speed [kph]",ax=axes[1,0])
plt.colorbar(im,label="Wind Speed [kph]",ax=axes[1,1])

axes[2,0].pcolormesh(lon,lat,spd[1],cmap='inferno',shading='Gouraud',
                     vmin=smin,vmax=smax)
axes[2,0].set_title("Annual Average Wind Speed at Level 1")
axes[2,0].set_ylabel("Degrees Latitude")
axes[2,0].set_xlabel("Degrees Longitude")
axes[2,1].pcolormesh(lon,lat,spd[0],cmap='inferno',shading='Gouraud',
                     vmin=smin,vmax=smax)
axes[2,1].set_title("Annual Average Wind Speed at Level 0")
axes[2,1].set_xlabel("Degrees Longitude")
plt.colorbar(im,label="Wind Speed [kph]",ax=axes[2,0])
plt.colorbar(im,label="Wind Speed [kph]",ax=axes[2,1])

fig.suptitle("Annual Average Wind Speeds at Different Layers")
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```



This makes it quite obvious why planes flying at cruising altitudes can sometimes benefit from a serious tailwind.

1.2 Vertical Slices

Let's say we don't care that much about the latitude-longitude data, but we do want to know how the atmosphere changes as a function of height and latitude.

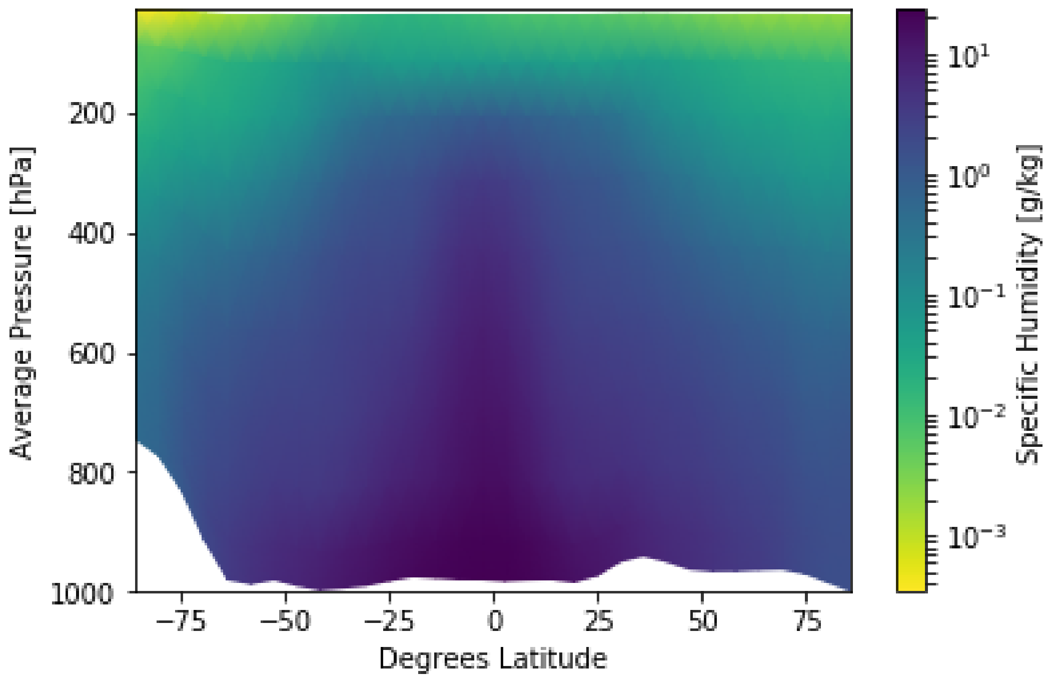
First it may be useful to define the pressure in every cell.

```
In [24]: pressure = np.zeros((12,10,32,64))
         ps = mydata.variables["ps"][:]
         lonlevs,llons = np.meshgrid(lev,lon,indexing='ij') #lonlevs will now have dimensions (1
         for m in range(0,12):
             for l in range(0,32):
                 pressure[m,:,l,:] = lonlevs*np.tile(ps[m,l,:],(10,1))
```

```
In [25]: x = np.mean(np.mean(mydata.variables['hus'][:],axis=0),axis=2)*1.0e3
         #Specific humidity [g/kg], averaged over time and longitude
         y = np.mean(np.mean(pressure,axis=0),axis=2)
         im=plt.pcolormesh(lat,y,x,cmap='viridis_r',shading='Gouraud',
```

```
                        norm=colors.LogNorm(vmin=np.amin(x),vmax=np.amax(x)))
plt.gca().invert_yaxis()
plt.ylabel("Average Pressure [hPa]")
plt.xlabel("Degrees Latitude")
plt.colorbar(im,label="Specific Humidity [g/kg]")
```

Out[25]: <matplotlib.colorbar.Colorbar at 0x7fd55d6fc8d0>

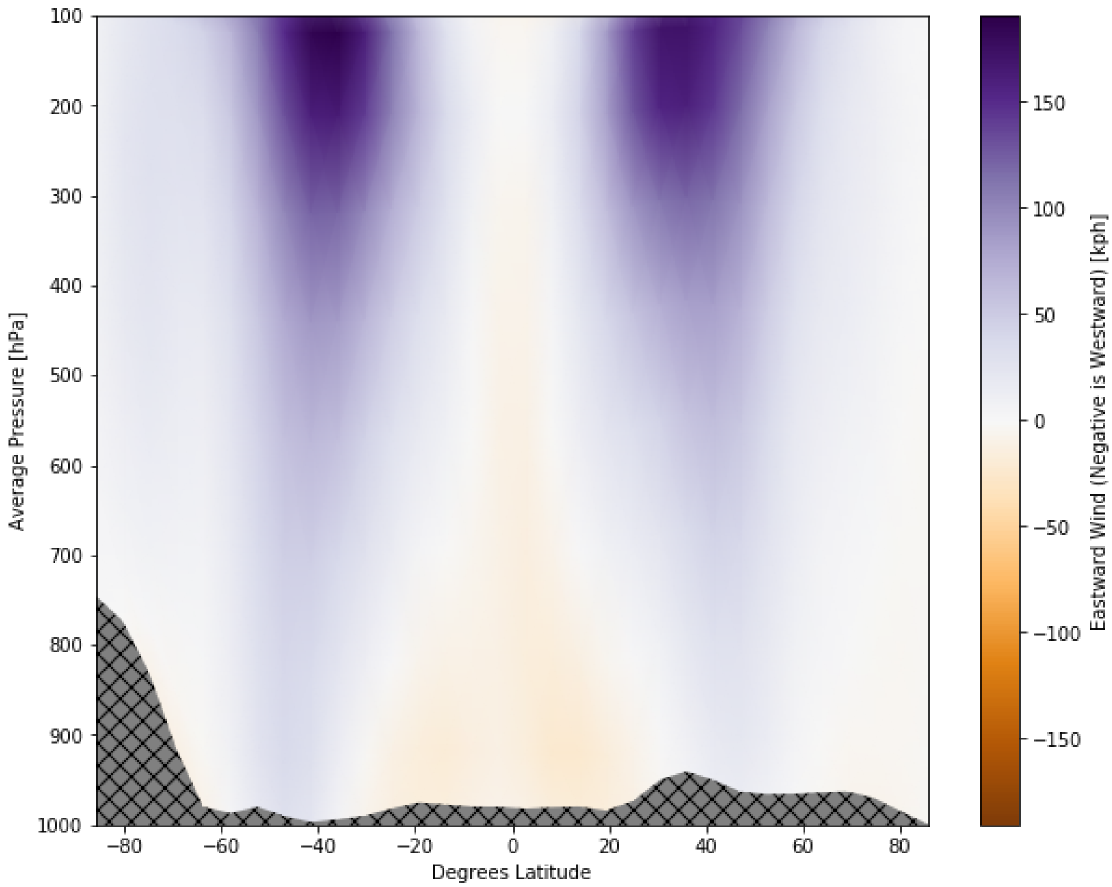


In [26]:
```
import matplotlib.patches as mpatches
```

In [27]:
```
x = np.mean(np.mean(mydata.variables['ua'][:],axis=0),axis=2)*3.6 #Eastward Wind [kph]
xmin = -np.amax(abs(x))
xmax =  np.amax(abs(x))

fig,ax=plt.subplots(figsize=(10,8))

p = mpatches.Rectangle((np.amin(lat),0),np.amax(lat)-np.amin(lat),1000,
                             hatch='xx', fill=True, facecolor='gray',zorder=-10)
ax.add_patch(p)
ax.patch.set_edgecolor('black')

im=ax.pcolormesh(lat,y,x,cmap='PuOr',shading='Gouraud',vmin=xmin,vmax=xmax)
ax.set_ylim(100,1000)

plt.gca().invert_yaxis()
```

```
ax.set_ylabel("Average Pressure [hPa]")
ax.set_xlabel("Degrees Latitude")
plt.colorbar(im,label="Eastward Wind (Negative is Westward) [kph]")
```

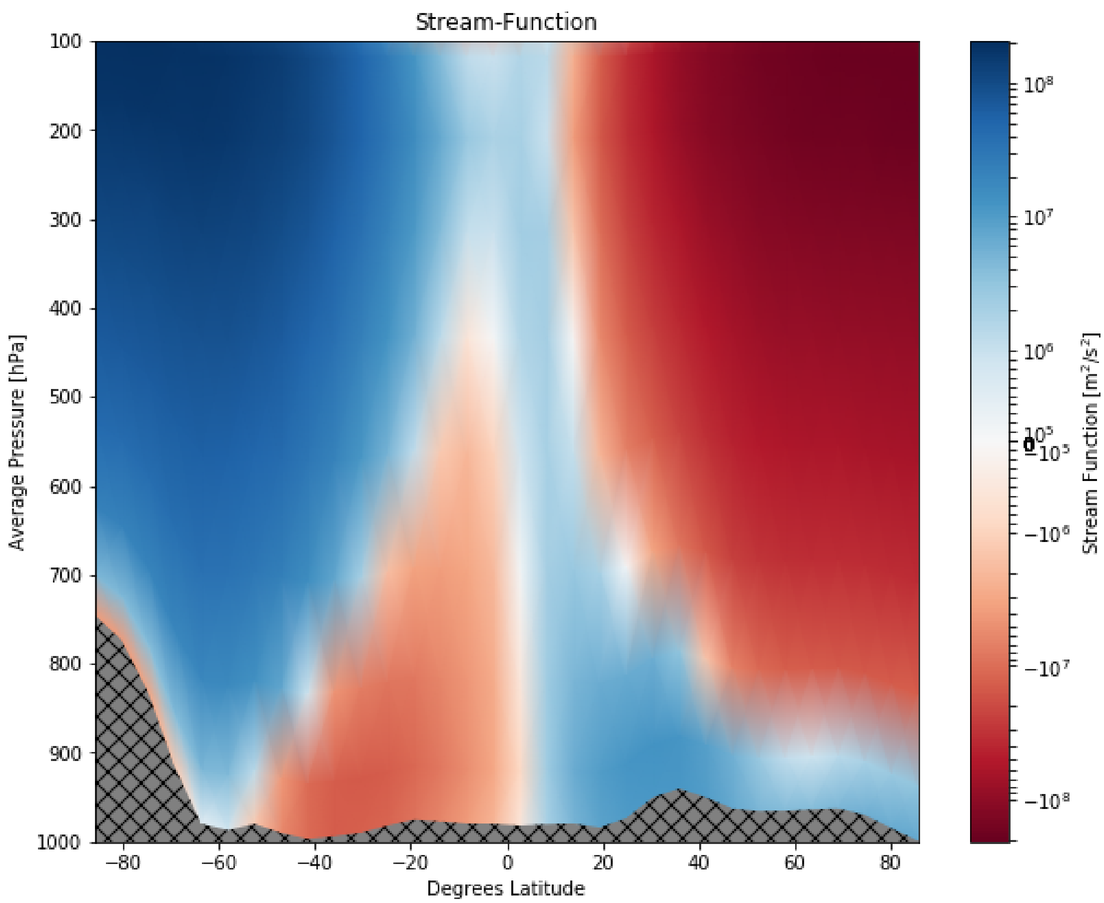
Out[27]: <matplotlib.colorbar.Colorbar at 0x7fd55d551950>



In [28]:
```
x = np.mean(np.mean(mydata.variables['stf'][:],axis=0),axis=2) #Stream-function [m^2/s^
xmin = -np.amax(abs(x))
xmax =  np.amax(abs(x))

fig,ax=plt.subplots(figsize=(10,8))

p = mpatches.Rectangle((np.amin(lat),0),np.amax(lat)-np.amin(lat),1000,
                       hatch='xx', fill=True, facecolor='gray',zorder=-10)
ax.add_patch(p)
ax.patch.set_edgecolor('black')

im=ax.pcolormesh(lat,y,x,cmap='RdBu',shading='Gouraud',vmin=xmin,vmax=xmax,
                 norm=colors.SymLogNorm(linthresh=1.0e6, linscale=1.0,vmin=-1.0, vmax=1
```

```
ax.set_ylim(100,1000)

plt.gca().invert_yaxis()
ax.set_ylabel("Average Pressure [hPa]")
ax.set_xlabel("Degrees Latitude")
plt.colorbar(im,label="Stream Function [m$^2$/s$^2$]")
plt.title("Stream-Function")
```

```
Out[28]: Text(0.5,1,u'Stream-Function')
```

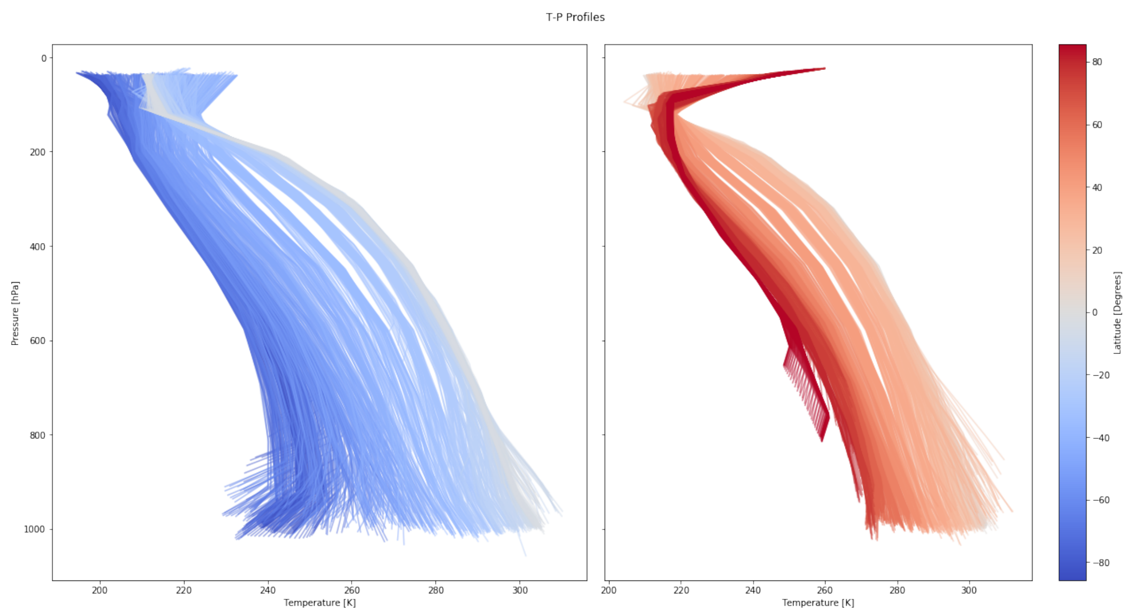


1.3 Other Kinds of Analysis

It may also be useful to plot something like the temperature-pressure profile, perhaps as a function of latitude.

```
In [29]: x = mydata.variables['ta'][11,:,:,:] #December
         clats = plt.cm.coolwarm(np.linspace(0.0,1.0,num=32))
         im=plt.scatter(np.arange(32),np.arange(32),
                        c=np.linspace(np.amin(lat),np.amax(lat),num=32),cmap="coolwarm")
```

```
plt.close('all')
fig,axes=plt.subplots(1,2,squeeze=True,sharey=True,figsize=(18,10))
for jlat in range(0,16):
    for jlon in range(0,64):
        axes[0].plot(x[:,jlat,jlon],pressure[11,:,jlat,jlon],
                         color=clats[jlat],alpha=0.5)
for jlat in range(16,32):
    for jlon in range(0,64):
        axes[1].plot(x[:,jlat,jlon],pressure[11,:,jlat,jlon],
                         color=clats[jlat],alpha=0.5)
axes[0].set_xlabel("Temperature [K]")
axes[0].set_ylabel("Pressure [hPa]")
axes[1].set_xlabel("Temperature [K]")
plt.gca().invert_yaxis()
plt.colorbar(im,label="Latitude [Degrees]",ax=axes[1])
plt.suptitle("T-P Profiles")
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```



1.4 A Note on Map Projections

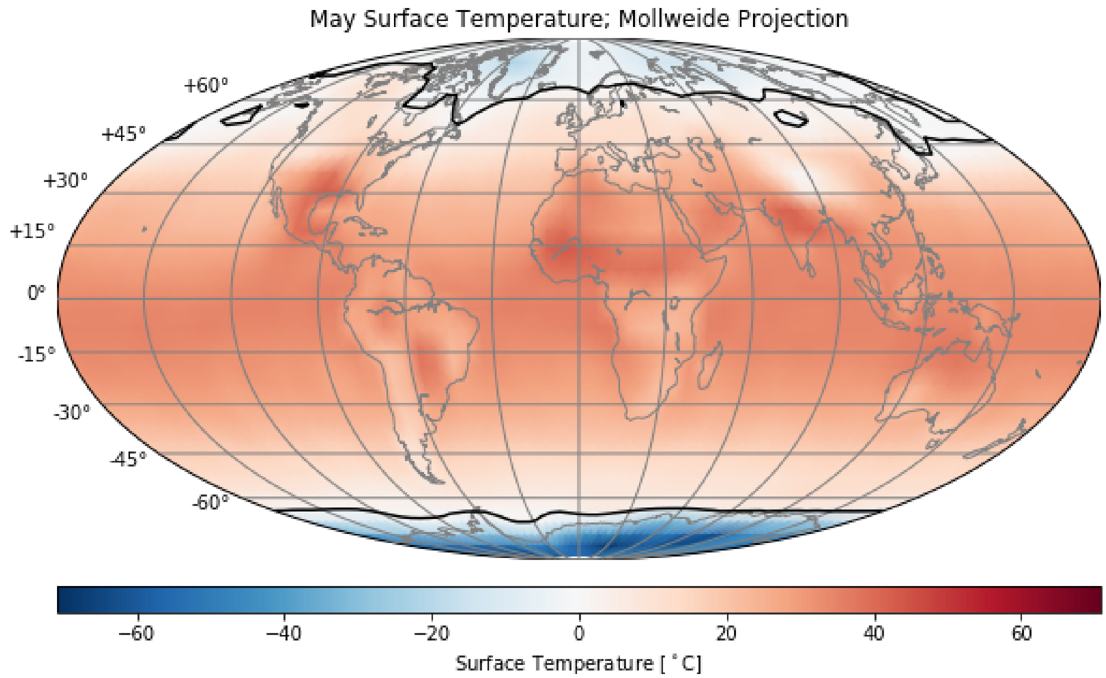
All of our plots thus far have been done on a rectlinear latitude-longitude grid, with equally-spaced latitudes and longitudes. This exaggerates higher latitudes, because the distance between two meridians decreases as a function of latitude. To represent actual land area more faithfully, you need to use a different map projection. Some projections are better than others, depending on what you want to show. The best for general use, in my opinion, is the Mollweide projection. It does a good job of preserving both shape and area.

You can use different projections through the Basemap package (https://matplotlib.org/basemap/). Installation is not entirely painless, but it's not necessarily the worst, either. If you can get it working, you can take some extra steps to plot using Mollweide projections:

```
In [30]: from mpl_toolkits.basemap import Basemap
```

We need one more longitude, so that variables wrap around and include both 0 and 360 degrees.

```
In [31]: def wrap2d(variable): #only input latitude-longitude arrays--2D only!
             newz = np.zeros(np.array(variable.shape)+np.array((0,1)))
             newz[:,:-1] = variable[:,:]
             newz[:,-1] = variable[:,0]
             return newz

In [32]: lons,lats = np.meshgrid(lon,lat) #Make 2D
         lonsw = wrap2d(lons)
         lonsw[:,-1] = 360.0 #Instead of 0.
         latsw = wrap2d(lats)

In [33]: x = wrap2d(mydata.variables["ts"][4,:,:]) - 273.15 #May
         tmin = -np.amax(abs(x))
         tmax =  np.amax(abs(x))

         fig,ax = plt.subplots(figsize=(10,8))

         m=Basemap(projection='moll',resolution='c',lon_0=0)
         im=m.pcolormesh(lonsw,latsw,x,cmap='RdBu_r',shading='Gouraud',
                         latlon=True,vmin=tmin,vmax=tmax)
         m.contour(lonsw,latsw,x,(0,),colors='k',zorder=3,latlon=True)
         m.drawcoastlines(color='gray')
         parallels = np.arange(-60.,61,15.)
         # labels = [left,right,top,bottom]
         m.drawparallels(parallels,labels=[True,False,False,False],
                         dashes=[1,0],color='gray',labelstyle="+/-")
         meridians = np.arange(-150,151,30.)
         m.drawmeridians(meridians,labels=[False,False,False,False],dashes=[1,0],color='gray')
         #Can't label meridians in a Mollweide projection
         m.colorbar(im,location='bottom',pad='5%',label='Surface Temperature [$^\circ$C]')
         plt.title("May Surface Temperature; Mollweide Projection")

/usr/local/lib/python2.7/dist-packages/mpl_toolkits/basemap/__init__.py:1707: MatplotlibDeprecat
  if limb is not ax.axesPatch:


Out[33]: Text(0.5,1,u'May Surface Temperature; Mollweide Projection')
```


May Surface Temperature; Mollweide Projection
+60°
+45°
+30°
+15°
0°
-15°
-30°
-45°
-60°
−60
−40
−20
0
20
40
60
Surface Temperature [˚C]