Processing Binary Data

Python and Binary Data

Many satellite data sets are distributed as binary files, often in compressed one or two byte formats. Usually software for reading the data is provided, but in some cases a simply python script can do the job better. Attached are some use cases for satellite data processing, identified by satellite or retrieval product.

GSMaP

In this use case a python script reads binary GSMaP data into well-formed arrays which then are written to a netcdf file. The GSMaP data is hourly and on a 0.1deg grid between 60N and 60S. It masks the GSMaP -99.0 missing value and converts the output to CF compliant naming and units. Here two months of data are processed. Some of the paths to the data will be depreciated.

```
unit_conversion = 1./3600.
gsm_out = '/work/mh0492/m219063/DYAMOND/GSMaP.v7_0.10deg.nc'

if (os.path.isfile(gsm_out)):
    print (gsm_out+' exists, not overwriting')
else:
    time= pd.date_range(ym+'2016-08-01','2016-09-30', freq='1H')
    lat    = xr.Coordinate('lat',
np.arange(-59.95,60.,0.1)[::-1].astype('float32'),
attrs={'long_name':'latitude','standard_name':'latitude','units':
"degrees_north"})
    lon    = xr.Coordinate('lon', np.arange(
0.05,360,0.1).astype('float32'),
attrs={'long_name':'longitude','standard_name':'longitude','units':
"degrees_east"})
    da     = np.ndarray(shape=(time.size,lat.size,lon.size))

    path =
'/work/mh0492/m219063/DYAMOND/Data/GSMaP/standard/v7/hourly/2016/'
    for i in np.arange(time.size):
        gsm_in = path + time[i].strftime("%m/%d")+
'/gsmap_mvk.2016'+time[i].strftime("%m%d.%H")+'00.v7.0001.0.dat'
        if (os.path.isfile(gsm_in)):
            with open(gsm_in,'rb') as f:
                data = np.fromfile(f, dtype=np.float32, count =
lon.size*lat.size)
            d2   = pd.DataFrame(np.reshape(data,(lat.size,lon.size)), lat,
lon)
            da[i,:,:] = d2.where(d2 != -99.0) * unit_conversion
        else:
            print(gsm_in)
```

```
    ds     = xr.Dataset({'pr': (['time', 'lat', 'lon'],
da.astype('float32'))},
                        coords={'time': time,'lat': lat,'lon': lon})

    ds.pr.attrs['long_name']    ='precipitation'
    ds.pr.attrs['standard_name']='precipitation_flux'
    ds.pr.attrs['units']        ='km m-2 s-1'
    ds.to_netcdf(gsm_out)
```

GMI

This example works with Remote Sensing Systems single byte binary files. Here GMI 3 day averaged data files are read and their data covered to NetCDF. For lack of a better method and because this was a one-time task, I used a rather inefficient (looping over ord) to convert ascii byte characters to integers for rescaling; ifield specifies which data field to process with possible fields being SST (0), Wind – low frequency (1), Wind – high frequency (2), precipitable water (3), cloud water (4) and rain (5).

```
gmi_out = '/work/mh0492/m219063/DYAMOND/Data/GMI-PRW_0.25deg.nc'
xscale = np.asarray([ 0.15, 0.2, 0.2, 0.3, 0.01, 0.1])
xoffset= np.asarray([-3.  , 0. , 0. , 0. ,-0.05, 0. ])
xfact = 251
ifield = 3
unit_conversion = 1.0

gmi_time= pd.date_range('2016-08-01','2016-09-30', freq='1d')
lat     = xr.Coordinate('lat',
np.arange(-89.875,90.,0.25)[::-1].astype('float32'),
attrs={'long_name':'latitude','standard_name':'latitude','units':
"degrees_north"})
lon     = xr.Coordinate('lon', np.arange(
0.125,360,0.25).astype('float32'),
attrs={'long_name':'longitude','standard_name':'longitude','units':
"degrees_east"})
da      = np.ndarray(shape=(gmi_time.size,lat.size,lon.size))

if (os.path.isfile(gmi_out)):
    print (gmi_out+' exists, not overwriting')
else:
    for i in np.arange(gmi_time.size):
        gmi_in = '/work/mh0492/m219063/DYAMOND/Data/GMI/f35_2016' +
gmi_time[i].strftime("%m%d")+'v8.2_d3d'
        print (gmi_in)
        with open(gmi_in,'rb') as f:
            dx = np.fromfile(f, dtype='S1',count=-1)

        i1 = lon.size*lat.size * ifield
```

```
        i2 = i1 + lon.size*lat.size
        data = np.ndarray(lon.size*lat.size)

        for j, x in enumerate(dx[i1:i2]):
            if (len(x) != 0): data[j] = ord(x)

        d2        = pd.DataFrame(np.reshape(data,(lat.size,lon.size)), lat,
lon)
        da[i,::-1,:] = (d2.where(d2 < xfact) * xscale[ifield] +
xoffset[ifield]) * unit_conversion

    if (ifield == 3):
        ds        = xr.Dataset({'prw': (['time', 'lat', 'lon'],
da.astype('float32'))},
                               coords={'time': gmi_time,'lat': lat,'lon':
lon})

        ds.prw.attrs['long_name']    ='precipitable water (vapor)'
ds.prw.attrs['standard_name']='atmosphere_mass_content_of_water_vapor'
        ds.prw.attrs['units']        ='kg m-2'
        ds.prw.attrs['source']       ='compiled from v8.2 d3d binary data
files provided by REMSS'

        plt.plot(ds.prw[:,:,:].mean(dim=('time','lon')))

        ds.to_netcdf(gmi_out)
```